Indonesian Journal of Electrical Engineering and Computer Science
Vol. 30, No. 3, June 2023, pp. 1586∼1595
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v30.i3.pp1586-1595

Improving the efficiency of clustering algorithm for duplicates detection

Abdulrazzak Ali[1], Nurul Akmar Emran[1], Safiza Suhana Kamal Baharin[1], Zahriah Othman[1], Awsan Thabet Salem[2], Maslita Abd Aziz[1], Nor Mas Aina Md Bohari[1], Noraswaliza Abdullah[1]
[1]Computational Intelligence Technologies (CIT) Research Group, Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Durian Tunggal, Malaysia
[2]Faculty of Computer and Information technology, Aden University, Aden, Yemen

Article Info

Article history:

Received Jul 5, 2022
Revised Jan 27, 2023
Accepted Jan 27, 2023

Keywords:

Attribute selection
Clustering data
Duplicate detection
Missing values
Sort key

ABSTRACT

Clustering method is a technique used for comparisons reduction between the candidates records in the duplicate detection process. The process of clustering records is affected by the quality of data. The more error-free the data, the more efficient the clustering algorithm, as data errors cause data to be placed in incorrect groups. Window algorithms suffer from the window size. The larger the window, the greater the number of unnecessary comparisons, and the smaller the window size may prevent the detection of duplicates that are supposed to be within the window. In this paper, we propose a data pre-processing method that increases the efficiency of window algorithms in grouping similar records together. In addition, the proposed method also deal s with the window size problem. In the proposed method, high-rank attributes are selected and then preparators are applied to the selected traits. A compensation algorithm is implemented to reduce the problem of missing and distorted sort keys. Two datasets (compact disc database (CDDB) and MusicBrainz) were used to test duplicates detection algorithms. The duplicates detection toolkit(DuDe) was used as a benchmark for the proposed method. Experiments showed that the proposed method achieved a high rate of accuracy in detecting duplicates. In addition, the proposed method.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Nurul Akmar Emran
Computational Intelligence Technologies (CIT) Research Group
Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka
Ayer Keroh, 76100 Durian Tunggal, Melaka, Malaysia
Email: nurulakmar@utem.edu.my

1. INTRODUCTION

It is undeniable that data are important assets for many organizations. While having high quality data is always desirable, data are often dirty at its source. Dirty data contain errors such as duplicate records which makes duplicates detection imperative [1].

The steady growth of datasets makes duplicates detection expensive. Online shops, as an instance, offer large catalogs comprising a constantly increasing set of gadgets from many distinctive suppliers. As impartial humans change the product portfolio, duplicates arise [2]. Similarly online applications suffer from the problem of duplicates data points. As a result, Facebook loads 10 terabytes of new data per day, and Yahoo graph stores 1.4 billion knots and 6.6 billion edges [3]. Thus, this undesirable data redundancy caused by duplicates data leads to an increase of accumulated digital universe data from 4.4 zettabytes (or trillion GB) in late 2015 to 44 zettabyte by 2024 [4]. Several studies have attempted to improve detection of duplicates where several terms such as merge/purge, record linkage, data matching, and object matching, are used to refer to the effort. Based on comprehensive comparisons of all pairs of records within a dataset to detect duplicates with naïve method showed quadratic complexity and also impractical with large datasets [5]. In this approach, the number of comparisons is time-consuming, for the average and large-sized datasets. In dealing with large datasets, the time taken for duplicates detection becomes an important factor. Therefore, techniques that can reduce the number of pairs of candidates for comparison such as standard blocking and sorted neighborhood are desirable as they depend on candidate keys to sort datasets and block (or windows) to organize datasets [6]. In general, clustering methods aim to raise the efficiency of duplicates detection by making comparisons confined between records that are likely to be similar and thus eliminating unnecessary comparisons.

Duplicate occurs if a unique entity object is represented several times [7]-[9] or the objects mirror the identical real-world object but have a variety of representations in the database [10], [11]. Duplicate detection involves comparing database objects, which can be costly with big datasets. Hashing datasets into small portions is known as clustering [12], and it is a common approach in data science in dealing with massive datasets. Similarity coefficient is a measure used in clustering data into small groups where similar records are placed in the same group [13]. In this section, we present a brief overview of standard clustering methods.

2. CLUSTERING OVERVIEW

Initially, detection of duplicate records is often based on comparison of records in the database where each record is compared with all records in the database. In the case of detecting duplicate records in two sets of data, each record in the first set is compared with all the records of the second group. For example, suppose that we have two datasets (M,N), where each contains 1,000,000 records, then; each record in group [M] will be compared to all the records of the group [N] so that the number of comparisons for both groups is [M]x[N] (10^{12}) this method is known as the naive method. The naive method for detecting duplicates relies on a quadratic number of pairwise comparisons of records to determine the duplicates. This number of comparisons can take hours even for a medium-sized dataset. Since today's databases are growing very rapidly, various clustering methods, such as standard blocking method, sorted neighborhood and canopy clustering address this problem by shrinking the number of comparisons.

2.1. Standard blocking

Standard blocking (SB) method uses the block key value to sort the records and group them into separate blocks where each block has the same key (BKV). A sort key consists of a single attribute or the combination of values of several attributes of a dataset where all records within the same block are compared. If the detection of duplicate records is necessary in the dataset, the R and B records represent the number of blocks given by the block method. Therefore, the number of pairs of records is specified as: $O(\frac{R2}{B})$ for comparison. This method elevates the comparison speed because of the limited number of comparison between records within the same block.

The weakness of this method is the records are disseminated in equal blocks size, thus, false results can be obtained by increasing the number of mismatches. This is due to the non-comparison of records that do not belong to the same block. Hence, it is necessary to choose appropriate keys for sorting so that the error ratio can be reduced and records on the blocks can be distributed equally.

2.2. Sorted neighborhood

Sorted neighborhood (SNM) method was proposed by Hernandez and Stolfos in 1998, (which is also called windowing) that aims to reduce the in steps in comparison process [14]. This method represents an advanced step in the sorting method as compared to the basic way, where the records within the window are characterized as high matching ratio. In the single neighborhood (SN) technique, it is more efficient to generate a sort key from multiple selected attributes than to rely on a specific attribute as it limits false classification of potential candidates as similar records.

Next, the comparison of records within the same window is conducted, where the number of comparison pairs is $n(n-1)/2$. When a comparison cycle is completed, the record at the top of the window slides out, the next record for the last record is added to the window and a new comparison cycle takes place. The fixed window size is denoted by w, so any new record that enters the window is compared to the previous one $|w|-1$.

The SN method accuracy relies on the key and window size quality. Fixed sliding window size presents a challenge with this method because a large window size may cause a large number of unnecessary comparisons and vice versa, and setting a small window size may override a number of potential duplicates [15]. Hence, the key quality and window size are the challenges in SN technique. This problem can be resolved by executing the technique independently several times and the different key is used to sort record each time with a small window size. The final result is based on the integration of results where this method is known as a multi-pass SN as proposed by [16].

In fact, both techniques SB and SNM have commonalities as both techniques assume that records that are close to each other have the potential to be duplicated [17]. In both methods, there is a cost-benefit trade-off caused by the size of the blocks or window. If blocks or the window contains a large number of records, a large number of comparison pairs will be generated.

2.3. Bigram indexing

The bigram indexing (BI) method was proposed in record linkage [18]. This method is less sensitive on missing information and typographical errors as compared to SB and SNM. The basic idea of this technique is that the sorting key is converted to a list representing sub-strings where each sub-string is composed of two characters so that all possible permutations are taken to form a list of Bigrams and the list is sorted and inserted into an inverted index. The number of sub-lists formed for a blocking key value relies on the value and threshold length. The shorter the sub-lists, the lower the threshold, yet, the longer the sub-lists per blocking key value, the smaller blocks in the inverted index.

2.4. Canopy clustering blocking

This method is called canopy clustering as proposed by McCallum *et al.* [19] in 2000. Canopy clustering is created by selecting a record from the set of records randomly. Each record is considered a center point where records are added to the randomly chosen record cluster, which is at a loose threshold distance to form the canopy clustering. In this method, the candidate records keys are created using Bigrams as tokens where the probability of error is low in order to create the inverse index table. The value of the threshold is calculated for records, either its value is close to the threshold value of the center or the value of the threshold is far from the center threshold value. The threshold value is determined by the user or from past experiences.

The number of record pairs comparisons as a result of canopy clustering is $O(\frac{fn^2}{c})$ [19]. Where n represents the number of records in each of the two dataset, c represents the number of canopies and f represents the average number of canopies a record belongs to. Therefore, if f is too small, this method is unable to detect typographical errors. Nevertheless, both bigram and canopy are classified as adaptive sorted neighborhood methods that are based on the assumption that the distance between the sorted records (ie. the record and the next) increases monotonously within a small neighborhood even though the sorting is done lexicographically rather than by distance [20].

As long as clustering data into small groups to avoid unnecessary comparisons depends on the similarity of the attribute(s) values used as sort keys, the failure is mainly due to the invalidity of the sort keys used. The ineffectiveness of sort keys are due to data errors (such as input errors, typos, misspellings, or missing values) in the attribute values or attributes used as sort keys. However, there is a need for prior processes to overcome the problem of distorted sorting keys that lead to the failure of grouping similar records in the same groups. Application of standard data preparation procedures plays a role in raising the similarity rate and eliminating differences arising due to input problems [21].

Missing values is a common issue in the existing data clustering methods [22]. The problem of missing values is usually dealt with either by ignoring those values, deleting incomplete records, or using one of the methods of imputation [23]. The first two options are impractical, thus compensating for missing values by using estimated values based on the available data within the dataset is commonly adopted. To improve the efficiency of the collection methods, the data preparation stage is followed by compensating the missing values using one of the clustering methods to deal with the incompleteness problem [24]. The impact of missing values on data clustering were discussed in [25].

In the next section, the proposed strategy for improving the effectiveness of data clustering methods will be presented. The duplicate detection model in [24] was used to measure the efficiency of the procedures adopted on the sorted neighborhood algorithm. In this paper, we aim to shed light on the measures that contribute to improve the effectiveness of clustering methods. These procedures focus on overcoming the factors affecting the validity of the sorting keys. The rest of this paper is structured as the following: section 2 consists of the details of the proposed method and the experiment setup. Section 3 is where the results and discussion are elaborated. Section 4 concludes the paper.

3. METHOD

In this section, we describe the proposed preprocessing steps which are (attributes selection, missing values compensation, comparative strings, and dynamic sorting keys). These steps aim to improve the performance of the clustering algorithms by reducing the difference gap between similar records. Figure 1 shows the proposed method structure.

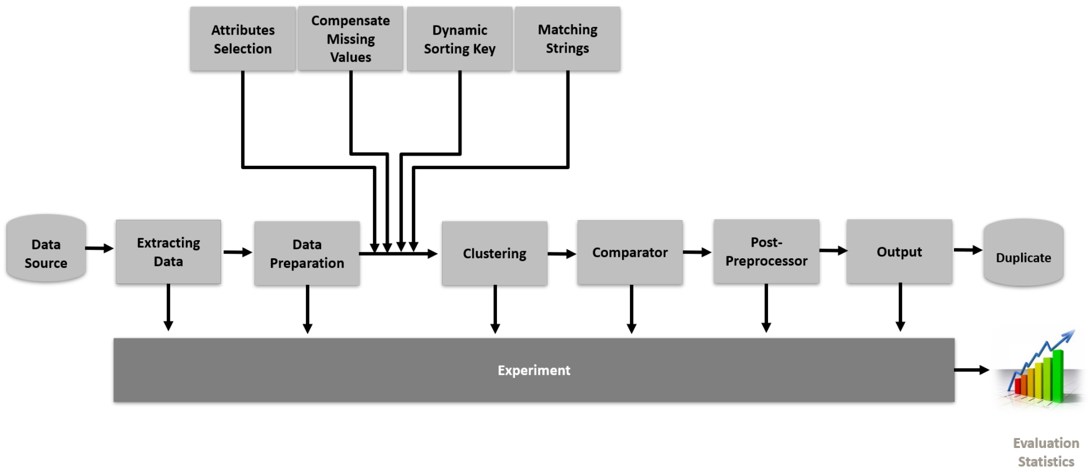


Figure 1. Method structure

3.1. Data preparation

Data preparation is the process of purifying and transforming raw data into a format that makes it fit to use. In data clustering, data preparation is an important and delicate task to narrow the gap between similar data within the dataset and those affected by errors [26]. Errors make identical data look different. Therefore, as long as the collection methods depend on the similarity measure in clustering the data, it is necessary to apply a minimum data setting to increase the similarity rate between the attribute(s) values.

Unfortunately, it is hard to find a standard number or sequence of data preparation processes [21]. For example, some attributes such as (phone number or date) need a standard format while other attributes may need other preparators. Thus, the dataset must be examined to determine the appropriate preparators. In this paper, minimal data preparation is applied to the attributes of the datasets under study such as removing special characters, attributes format standardization, UTF−8 to ASCII, capitalize, dyllabify, and dtem.

3.2. Attributes selection

Sort keys usually consist of the values of a single attribute or of combination of values of attributes together into a single string [24], [27]. Therefore, the process of identifying the attribute(s) of the sort key plays a crucial role in clustering records that are likely to be similar in the same group. In duplicates detection, primary keys within datasets are often avoided in duplicate detection because the keys give a false impression that there is no duplication.

The attribute selection stage is heuristic for high-ranking attributes that will be used later in the clustering and matching stages. This stage plays a major role in generating strong sort keys that increases the probability of detecting true duplicates, as well as avoiding attributes that give a false sense of duplicate as attributes (gender or zip-code). Also, it reduces the cost of detecting duplications by making the matching process among the selected attribute values only instead of comparing all the attributes values of the dataset (for more see [25]).

3.3. Missing values compensation

Missing values problem is an obstacle in achieving an optimal performance of clustering algorithms. Therefore, a common practice in dealing with the problem of missing values within the context of clustering is by compensating the missing values before executing the clustering algorithm against the complete data [22].

Compensation algorithms are effective as missing values in records are replaced by values obtained from related cases in other records within the dataset, as in the case with the Hot Deck algorithm. Missing values can be replaced with values close to the real value using the Nearest Neighbor algorithms [28]. Therefore, missing values compensation is a critical process to overcome the problem of distorted or missing sort keys, which reduce the efficiency of clustering methods. Because we aim to improve the process of detecting duplicates, we adopt the hot deck compensation method, which depends on compensating the missing values from the closest similar record within the dataset. An experiment was conducted to test the hypothesis that the proposed preprocessing process can improve the accuracy of clustering algorithms.

3.4. Datasets

In the experiment, we used the compact disc database (CDDB) and MusicBrainz datasets. These datasets were provided by the Hasso-Plattner-Institute (HPI) which have been used frequently in duplicates detection research. To simulate data errors on a larger scale, the MusicBrainz dataset generated with the DaPo data pollution tool presented in [29] was used to test redundancy detection algorithms:

- CDDB dataset was extracted from the freeDB repository. This dataset consists of 9,763 real audio CDs each record contains information (such as artist, title, and tracks), which contain 299 duplicates records.
- MusicBrainz dataset contains 193,750 tuples, where each tuple represents a certain audio recording. The size of the generated duplicate clusters ranges from 1 to 5. For a further description of the contents of this dataset attributes, you can refer to [29].

To simulate incompleteness in the CDDB dataset, missing values of 4% were injected using a random pattern [24]. For the MusicBrainz dataset the percentage of missing values was already high in key attributes such as Title 0.67%, Artist 21% and Album 24%.

3.5. Experiment configuration

To show the effect of the proposed preprocessing on the clustering algorithms, the DuDe toolkit has been used. DuDe toolkit is available from the HPI. Because of the working mechanism of the $NaiveDuplicate-Detectionalgorithm$ that is based on comparing all the records of the dataset with each other, this is not feasible for large datasets due to the high computation cost. In addition, it was used only with the CDDB dataset to show the effect of the preprocessing procedures. For the $SortedNeighborhoodMethod$, we implemented the process five times with window size range from 50-250 records to show the effect of preprocessing stage on the window size drawback.

In the experiment, the Levenshtein Distance algorithm was used to obtain the duplicates records. The similarity threshold value was set to 0.9. Duplicates detection measures which are recall, precision and f-score were used to assess the accuracy of duplicates detection. The experiment was conducted in two stages to show the effect of the pre-processing stage on the efficiency of the work of the clustering algorithms. In the first stage, which is denoted by (before), the $artist$ attribute values for both datasets are used as sort keys, which are passed to the used clustering algorithm, while the $title$ attribute values are passed to the similarity metric. In the second stage, which is denoted by (after), the data is initially initialized by applying the data preparators, followed by the creation of sort keys from the values of the selected attributes according to the specified criteria (uniqueness and completeness). The sort keys in the second stage of the experiment consist of combining seven characters of the three first selected attribute (high-rank) values into a single string denoted by (SK). In the second stage of the experiment, the (SK) are passed to the clustering algorithm, and the same values are passed to the similarity metric. The declared duplicates were compared with the gold dataset by the statistical component where duplicate detection measures were used to evaluate the accuracy of both methods.

4. RESULTS AND DISCUSSION

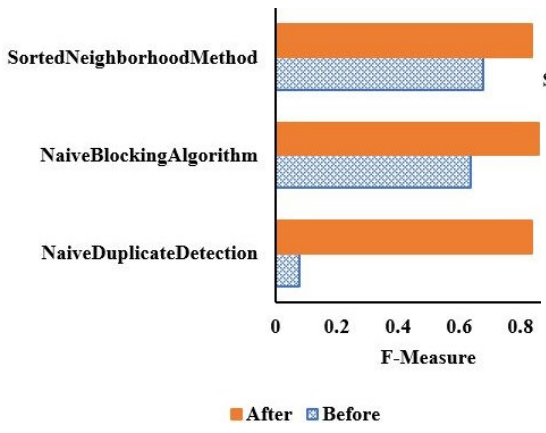
As long as the DuDe platform provides a wide range of clustering algorithms, is worth investigating whether the procedures adopted in the proposed method. The results of the experiments as shown in Table 1 and Table 2 for both datasets show that the clustering algorithms with the proposed preprocessing have achieved higher accuracy in duplicates detection with a lower error rate as compared to the results of the original DuDe. Figure 2 shows the improvement in the accuracy of the detection of duplicates using the preprocessing procedures for all the clustering algorithms within this study, especially CDDB dataset in Figure 2(a) and MusicBrainz dataset in Figure 2(b).

Table 1. Implementation results on CDDB dataset

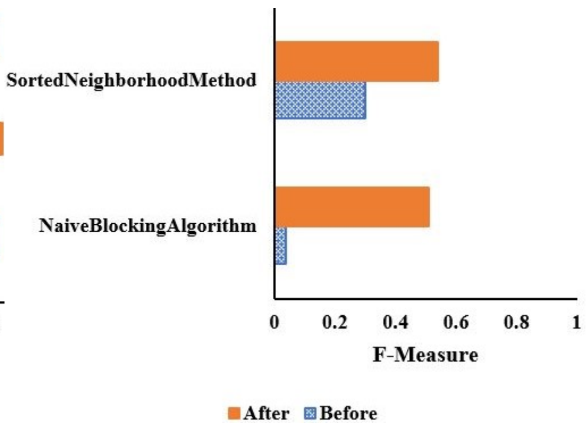
DuDe	Clustering method	Dup	TP	FP	FN
Before	NaiveDuplicateDetection	5,422	220	5,202	79
	NaiveBlockingAlgorithm	320	197	123	102
	SortedNeighborhoodMethod	306	205	101	94
After	NaiveDuplicateDetection	262	236	26	63
	NaiveBlockingAlgorithm	234	228	6	71
	SortedNeighborhoodMethod	262	236	26	63

Table 2. Implementation results on MusicBrainz dataset

DuDe	Clustering method	Dup	TP	FP	FN
Before	NaiveBlockingAlgorithm	1,201,486	29,357	1,172,129	114,393
	SortedNeighborhoodMethod	59,757	30,737	29,020	113,013
After	NaiveBlockingAlgorithm	66,052	53,029	13,023	90,721
	SortedNeighborhoodMethod	79,134	60,438	18,696	83,312



(a)



(b)

Figure 2. Accuracy comparison of clustering algorithms: (a) CDDB and (b) MusicBrainz dataset

The results of the experiments in Tables 3 and 4 for datasets CDDB and MusicBrainz respectively show that the proposed model has achieved higher accuracy in duplicate detection with a lower error rate as compared to the results of the original DuDe in all implementation cycles. From the results, it was discovered that as the larger the window size was used, the greater the number of false positives for the original DuDe. This is because, the error indicators were influenced by data errors. For the original DuDe, the presence of spelling and syntax errors made the similar values appear different, which increased the value of false negatives, and the clustering of missing keys caused an increase in false positives. The proposed actions limited the impact of the errors and the window size drawback for detecting duplicates. Despite the use of different window sizes, the proposed method showed stability in the duplicates detection accuracy against the orginal DuDe. Figure 3 shows the effect of window size on the SNM algorithm using CDDB (Figure 3(a)) and MusicBrainz (Figure 3(b)) datasets.

Table 3. Implementation results with different window sizes for CDDB dataset

SNM	Size (W)	No of pairs	Declared duplicates	TP	FP	FN	F-score
Before	50	477,162	306	205	101	94	0.68
	100	961,587	376	205	171	94	0.61
	150	1,443,512	434	205	229	94	0.56
	200	1,922,937	491	205	286	94	0.52
	250	2,399,862	548	205	343	94	0.48
After	50	477,162	262	236	26	63	0.84
	100	961,587	262	236	26	63	0.84
	150	1,443,512	262	236	26	63	0.84
	200	1,922,937	262	236	26	63	0.84
	250	2,399,862	262	236	26	63	0.84

Table 4. Implementation results with different window sizes for MusicBrainz dataset

SNM	Size (W)	NO of pairs	Declared duplicates	TP	FP	FN	F-score
Before	50	9,492,525	59,757	30,737	29,020	113,013	0.3
	100	19,176,300	85,357	30,861	54,496	112,889	0.27
	150	28,857,575	109,838	30,938	789,000	112,812	0.24
	200	38,536,350	133,699	30,995	102,704	112,755	0.22
	250	48,212,625	156,943	31,032	125,911	112,718	0.21
After	50	9,492,525	79,134	60,438	18,696	83,312	0.54
	100	19,176,300	82,030	60,585	83,165	0.54	
	150	28,857,575	82,444	60,635	21,809	83,115	0.54
	200	38,536,350	82,548	60,690	21,858	83,060	0.54
	250	48,212,625	82,629	60,734	21,895	83,016	0.54

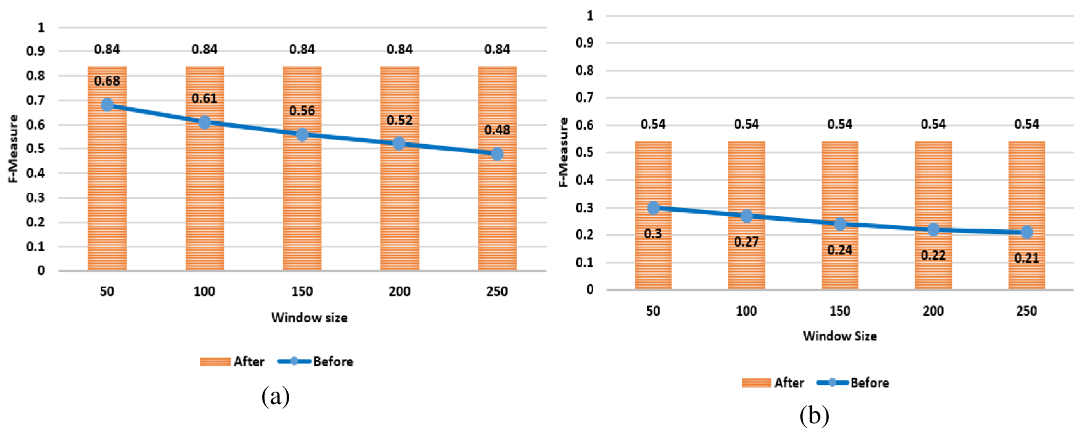


Figure 3. The accuracy of SNM on different window sizes: (a) CDDB and (b) MusicBrainz dataset

5. CONCLUSION

In conclusion, the efficiency of the clustering algorithm plays a major role in detecting duplicates by grouping the records that are likely to be similar together. This role is affected by the level of data quality within the dataset. In this paper, we focused on the pre-processing stage of the dataset and its impact on the clustering methods in duplicates detection. Experiments were conducted using real datasets to test the duplicate detection algorithms. The results showed that the pre-processing of the dataset improved the accuracy of duplicates detection. The results showed that the accuracy in the proposed method was not affected by the number of records within the window (by limiting the effect of window size on the detection accuracy) as it was in the original method DuDe.

ACKNOWLEDGMENT

We would like to thank Hasso-Plattner-Institute (HPI) and and Humburg university for the datasets and duplicate detection tool (DuDe) used this paper. We also acknowledge financial assistance received from the Universiti Teknikal Malaysia Melaka (UTeM) in publishing this work.

REFERENCES

[1] A. Samiei and F. Naumann, "Cluster-based sorted neighborhood for efficient duplicate detection," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016, pp. 202–209, doi: 10.1109/ICDMW.2016.0036.

[2] C. G. Prasad, M.Veena, and S. S. S. Chandra, "Usual duplicate detection using progressive algorithms with improving efficiency," *International Journal Magazine of Engineering, Technology, Management and Research*, vol. 4, no. 6, pp. 273-279, 2017.

[3] J.-Y. Lee, U. Kang, D. Koutra, and C. Faloutsos, "Fast anomaly discovery given duplicates," 2012, [Online]. Available: http://reports-archive.adm.cs.cmu.edu/anon/2012/CMU-CS-12-146.pdf.

[4] M. Kolanovic and R. Krishnamachari, "Big data and AI strategies: machine learning and alternative data approach to investing," 2017, [Online]. Available: https://www.cfasociety.org/cleveland/Lists/Events%20Calendar/Attachments/1045/BIG-Data_AI-JPMmay2017.pdf.

[5] L. Gagliardelli, S. Zhu, G. Simonini, and S. Bergamaschi, "Bigdedup: a big data integration toolkit for duplicate detection in industrial scenarios," *Advances in Transdisciplinary Engineering*, vol. 7, pp. 1015–1023, 2018, doi: 10.3233/978-1-61499-898-3-1015.

[6] C. Batini and M. Scannapieca, *Data quality*. Berlin: Springer, 2006, doi: 10.1007/3-540-33173-5.

[7] S. Moon and H. Ikeda, *Database systems for advanced applications '93*. World Scientific, 1993, doi: 10.1142/2010.

[8] J. J. Tamilselvi and C. B. Gifta, "Handling duplicate data in data warehouse for data mining," *International Journal of Computer Applications*, vol. 15, no. 4, pp. 7–15, Feb. 2011, doi: 10.5120/1939-2590.

[9] Q. Chen, J. Zobel, and K. Verspoor, "Benchmarks for measurement of duplicate detection methods in nucleotide databases," *Database*, 2017, doi: 10.1093/database/baw164.

[10] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, 2007, doi: 10.1109/TKDE.2007.250581.

[11] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger, and W. Schwinger, "'Same, same but different' a survey on duplicate detection methods for situation awareness," in *On the Move to Meaningful Internet Systems: OTM 2009. OTM 2009. Lecture Notes in Computer Science*, Berlin: Springer, 2009, pp. 1050–1068, doi: 10.1007/978-3-642-05151-7_22.

[12] S. M. Mohammed, K. Jacksi, and S. R. M. Zeebaree, "A state-of-the-art survey on semantic similarity for document clustering using GloVe and density-based algorithms," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 22, no. 1, pp. 552-562, 2021, doi: 10.11591/ijeecs.v22.i1.pp552-562.

[13] M. Fariss, N. El Allali, H. Asaidi, and M. Bellouki, "A semantic web services discovery approach integrating multiple similarity measures and k-means clustering," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 24, no. 2, pp. 1228-1237, 2021, doi: 10.11591/ijeecs.v24.i2.pp1228-1237.

[14] M. A. Hernández and S. J. Stolfo, "Real-world data is dirty: Data cleansing and the merge/purge problem," *Data Mining and Knowledge Discovery,* vol. 2, no. 1, pp. 9–37, 1998, doi: 10.1023/A:1009761603038.

[15] U. Draisbach and F. Naumann, "A generalization of blocking and windowing algorithms for duplicate detection," in *2011 International Conference on Data and Knowledge Engineering (ICDKE)*, 2011, pp. 18–24, doi: 10.1109/ICDKE.2011.6053920.

[16] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 127–138, 1995, doi: 10.1145/568271.223807.

[17] F. Naumann and M. Herschel, *An introduction to duplicate detection*. Cham: Springer International Publishing, 2010.

[18] P. Christen, "Febrl – Freely extensible biomedical record linkage," 2002, [Online]. Available: https://openresearch-repository.anu.edu.au/bitstream/1885/40723/3/TR-CS-02-05.pdf.

[19] A. McCallum, K. Nigam, and L. H. Ungar, "Efficient clustering of high-dimensional data sets with application to reference matching," *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2000, pp. 169–178, doi: 10.1145/347090.347123.

[20] S. Yan, D. Lee, M.-Y. Kan, and L. C. Giles, "Adaptive sorted neighborhood methods for efficient record linkage," in *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, 2007, pp. 185–194, doi: 10.1145/1255175.1255213.

[21] I. Koumarelas, L. Jiang, and F. Naumann, "Data preparation for duplicate detection," *Journal of Data and Information Quality*, vol. 12, no. 3, pp. 1–24, 2020, doi: 10.1145/3377878.

[22] S. Boluki, S. Z. Dadaneh, X. Qian, and E. R. Dougherty, "Optimal clustering with missing values," *BMC Bioinformatics*, vol. 20, no. S12, p. 321, 2019, doi: 10.1186/s12859-019-2832-3.

[23] A. Salem, N. A. Emran, A. K. Muda, Z. Sahri, and A. Ali, "Missing values imputation in Arabic datasets using enhanced robust association rules," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 28, no. 2, pp. 1067–1075, 2022, doi: 10.11591/ijeecs.v28.i2.pp1067-1075.

[24] A. Ali, N. A. Emran, and S. A. Asmai, "Missing values compensation in duplicates detection using hot deck method," *Journal of Big Data*, vol. 8, no. 1, p. 112, 2021, doi: 10.1186/s40537-021-00502-1.

[25] A. Ali, N. A. Emran, S. A. Asmai, and A. Thabet, "Duplicates detection within incomplete data sets using blocking and dynamic sorting key methods," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 9, 2018, doi: 10.14569/IJACSA.2018.090979.

[26] V. Devarajan and R. Subramanian, "Analyzing semantic similarity amongst textual documents to suggest near duplicates," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 25, no. 3, pp. 1703–1711, 2022, doi: 10.11591/ijeecs.v25.i3.pp1703-1711.

[27] D. Hadzic and N. Sarajlic, “Methodology for fuzzy duplicate record identification based on the semantic-syntactic information of similarity,” *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 1, pp. 126–136, Jan. 2020, doi: 10.1016/j.jksuci.2018.05.001.

[28] L. Beretta and A. Santaniello, “Nearest neighbor imputation algorithms: a critical evaluation,” *BMC Medical Informatics and Decision Making*, vol. 16, no. S3, p. 74, 2016, doi: 10.1186/s12911-016-0318-z.

[29] K. Hildebrandt, F. Panse, N. Wilcke, and N. Ritter, “Large-scale data pollution with Apache Spark,” *IEEE Transactions on Big Data*, vol. 6, no. 2, pp. 396–411, 2020, doi: 10.1109/TBDATA.2016.2637378.

BIOGRAPHIES OF AUTHORS

Mr. Abdulrazzak Ali Mohamed is Member of the Teaching Assistant at the College of Computer and Information Technology, University of Aden. He received a bachelor’s degree in computer programming from the University Of Mosul, College of Computer Sciences and Mathematics and a master’s degree in computer science from the Fakulti Teknologi Maklumat dan Komunikasi (FTMK), Universiti Teknikal Malaysia Melaka. During his work in the Programming, Research and Development Department of the Computer Center, he contributed to the design and development of a number of computer systems for the University of Aden. Currently, he is a Ph.D candidate in computer science at Information Technology from the Universiti Teknikal Malaysia Melaka. He can be contacted at email: dowsan1@yahoo.com.

Associate Professor Dr. Nurul Akmar Emran is an associate professor at the Universiti Teknikal Malaysia Melaka (UTeM). She received a bachelor’s degree in Management Information System (MIS) from the International Islamic University Malaysia in 2001, an MSc in Internet and Database Systems from London South Bank University in 2003, and a Ph.D. degree in computer science from the University of Manchester, the UK in 2011. She begins her career in academia in 2002, as a tutor at the Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka (UTeM). In 2004, she was appointed as a lecturer; in 2011, she was a senior lecturer before being promoted to Associate Professor in 2017. Her research interests include data quality (data completeness), database systems and security, storage space optimization, mobile analytics, and green computing. She can be contacted at email: nurulakmar@utem.edu.my.

Dr. Safiza Suhana Kamal Baharin works as a senior lecturer at Universiti Teknikal Malaysia Melaka (UTeM). She earned a bachelor’s degree in Science Geoinformatics (BSc. Geoinformatics (Hons)) from Universiti Teknologi Malaysia (UTM) in 2000 and a master’s degree in Science Geoinformatics by research from the same university. In 2016, she received her Ph.D. in Computer Science from Universiti Teknikal Malaysia Melaka (UTeM). In 2001. She started as a lecturer in 2002 and was promoted to senior lecturer in 2016. She teaches database, software engineering, programming technique and GIS courses at the undergraduate and post-graduate levels. Geographic information system (GIS), disaster management, spatial analysis, and spatial database are some of her current research interests. She can be contacted at email: safiza@utem.edu.my.

Dr. Zahriah Othman is a senior lecturer at the Universiti Teknikal Malaysia Melaka (UTeM). She received bachelor degree in Information Technology (BIT(Hons)) from the Universiti Utara Malaysia, in 2001, Msc in Software Engineering from Bradford University in 2003 and Ph.D. degree in computer science from Universiti Teknikal Malaysia Melaka in 2016. She begins her career in academia in 2002, as a tutor at the Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka (UTeM). In 2003, she was appointed as a lecturer before being promoted to senior lecturer in 2016. Currently, she is teaching software engineering and programming-related subjects at undergraduate and postgraduate levels. She holds Certified Testers Foundation Level (CTFL) in 2014 and her current research interests include information retrieval, software testing and GIS-related research. She can be contacted at email: zahriah@utem.edu.my

Mr. Awsan Thabet Salem He holds a bachelor's degree from Zarqa University in Jordan and a master's degree from the Universiti Teknologi MARA (UiTM), Malaysia. He worked as a computer instructor at the College of Computer and Information Technology, University of Aden. He can be contacted at email: awsanthabet666@gmail.com

Maslita Abd Aziz is a senior lecturer at Faculty of Information and Communication Technology, UTeM. She finished her first degree at Universiti Utara Malaysia (UUM) in 1995 with BSc in Information Technology (with Hons.) and later her MSc from Rochester Institute of Technology, New York, USA in 1998 with MSc in Information Technology (with Hons) specializing in Software Development and Management. Her research interests are in softcomputing, specifically on optimizing meta heuristic in software testing problems. She is also interested in improving teaching and learning approach of programming language especially for novice learner. She can be contacted at email: maslita@utem.edu.my

Nor Mas Aina Md Bohari is a lecturer at the Fakulti Teknologi Maklumat dan Komunikasi (FTMK), Universiti Teknikal Malaysia Melaka (UTeM). She received the Degree of Bachelor of Information Technology with Honours (BIT Hons) (major in Information Management) from Universiti Utara Malaysia (UUM) in 2002. She has been awarded the degree of Master of Science having followed an approved program in Object Oriented Information Systems in 2005 from London South Bank University, UK. She begins her career in academia in 2002, as a tutor at the FTMK, Kolej Universiti Teknikal Kebangsaan Malaysia (KUTKM). In 2005, she was appointed as a lecturer at FTMK, UTeM. She is also a member of the Malaysia Board of Technologists (MBOT) since 2018. Her key areas of interest are database design and development. She can be contacted at email: aina@utem.edu.my

Dr. Noraswaliza Abdullah is currently a senior lecturer in Faculty of ICT at the Universiti Teknikal Malaysia Melaka. She received her PhD from the Queensland University of Technology, Australia. Her work includes developing recommender system techniques by applying data mining techniques. Her research interests include data mining, recommender system and database technology. She can be contacted at email: noraswaliza@utem.edu.my.