International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 03 | March - 2024 SJIF Rating: 8.176 ISSN: 2582-3930

Data Duplication Removal Using Machine Learning

Nikhil Kumar .K , Ganesh Kumar .G , Hemanth Kumar .V , Jivesh Kumar , Vishal Kumar , Dinkar Kumar, M. Sathish Kumar

Student and Assistant Professor of[1]Excel Engineering College, NH-544, Salem Main Road, Sankari West, Pallakkapalayam, Pin:637 303. Komarapalayam Namakkal Dt.

[3]Anna University, Chennai ,Tamilnadu

Abstract

This paper introduces a novel concept termed near-duplicate datasets, which represent quasi-duplicate versions of original datasets subjected to unknown modifications such as row and column insertions and deletions. These datasets hold significant importance for various data-related tasks including exploration, integration, and quality assessment. The paper proposes an innovative method to detect near-duplicate datasets, leveraging features extraction and machine learning techniques. Unlike conventional approaches relying on direct column comparisons, this method focuses on comparing metadata vectors summarizing the datasets, thus offering a distinct and effective means of detection. Additionally, the paper presents a methodology for artificially generating training data to facilitate algorithm training. Through extensive experimentation, the paper identifies optimal parameters for training data creation and evaluates the performance of multiple classifiers. Notably, the results demonstrate an accuracy rate exceeding 95%, underscoring the efficacy of the proposed approach in effectively identifying quasi-duplicate datasets, thus offering valuable insights into data quality and integrity.

1. Introduction

In the contemporary post-digital era, data stands as a paramount asset for companies of all sizes, necessitating meticulous considerations for optimal storage solutions. However, despite its pivotal role in decision-making processes, a significant portion of global data repositories suffers from quality deficiencies, incurring substantial costs. Addressing this quality gap is imperative, and one fundamental aspect is identifying whether the required data already exists. While ample literature exists on near-duplicate document identification, the detection of nearly duplicated datasets, also known as fuzzy duplicated data, represents a distinct sub-domain within the realm of record linkage, also referred to as entity resolution. The classical process for detecting near-duplicate data involves several stages: data preparation, wherein data is enriched, cleaned, and standardized for enhanced

comparability; reduction of the search space to minimize comparisons; attribute-level comparison of candidate pairs using similarity measures; and decision-making on duplication status, employing either simple distance-based models or more sophisticated machine learning-based approaches. Subsequently, results are clustered to ensure consistency. In this paper, we introduce an original method leveraging machine learning that does not presume any prior knowledge of the data or attempt to identify its schema. Instead, each dataset is summarized as a metadata vector, focusing particularly on semi-structured data and "instance-level modifications." This research constitutes a component of a broader initiative aimed at metadata extraction in data lake architectures, with the ultimate goal of facilitating data integration, enhancing data quality, and streamlining data exploration processes.

2. Problem description

We define a dataset DS as an instance I of an unknown relational schema R, comprising K columns, where each record is termed a tuple t (a row). Each column C in the dataset is conceptualized as a multiset [2], denoted as C = (e1, e2, .., eL), containing L elements. Here, L represents the length of the column, and each e signifies an element of a multiset D, referred to as the domain. A near-duplicate dataset (NDDS) is characterized as a modified iteration of an existing dataset, where θ columns have been added, and ι columns have been deleted. We represent this transformation as $\zeta = 10 * \theta + \iota$ K. Furthermore, beta lines have been removed, and gamma lines have been inserted, denoted as $\alpha = \beta + \gamma$ L. The primary challenge lies in discerning whether a DS corresponds to a near-duplicate dataset, necessitating an understanding of the structural alterations, including column modifications and tuple additions or deletions.

3. Feature

To identify near-duplicate datasets, we employ a machine learning-based approach, necessitating the extraction of fixed-length feature vectors from each dataset. While Sherlock's method for Semantic Data Type Detection primarily focuses on column-level features, we adapted this technique for dataset-level analysis by concatenating each dataset into a single column. Several modifications were introduced to optimize this approach. Firstly, all datasets were preprocessed to lowercase, reducing the feature extraction complexity and enhancing learning efficiency. This transformation particularly benefits features reliant on character count, where each additional character exponentially increases feature count. By standardizing to lowercase, we streamline the extraction process without compromising accuracy. Secondly, we curated features from Sherlock, excluding those derived from word embedding, as they are irrelevant in our context. Additionally, while column length significantly

influences semantic type recognition in Sherlock, we intentionally disregard this feature to prevent column size from skewing decision-making. Given the ease of manipulating dataset length, we prioritize features independent of dataset size to compose our final feature vector. To illustrate the efficacy of our approach compared to Sherlock's, we conducted experiments drawing subsets of varying sizes from columns containing dates in YYYY format. The norm of feature vectors extracted using both methods was computed across subset sizes, revealing contrasting behaviors. While Sherlock's method exhibits norm growth proportional to column size, our method demonstrates minimal norm variation, particularly evident in smaller subsets. This distinction underscores the robustness and scalability of our approach in handling datasets of diverse dimensions. Figure 2 illustrates the features extraction process, showcasing our methodology's efficiency in capturing dataset characteristics essential for near-duplicate detection.

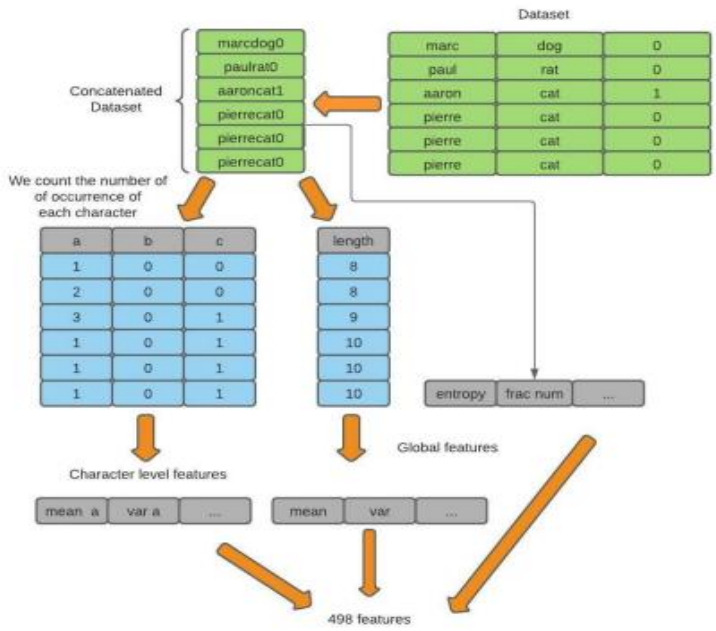


Figure. 1: Features extraction process

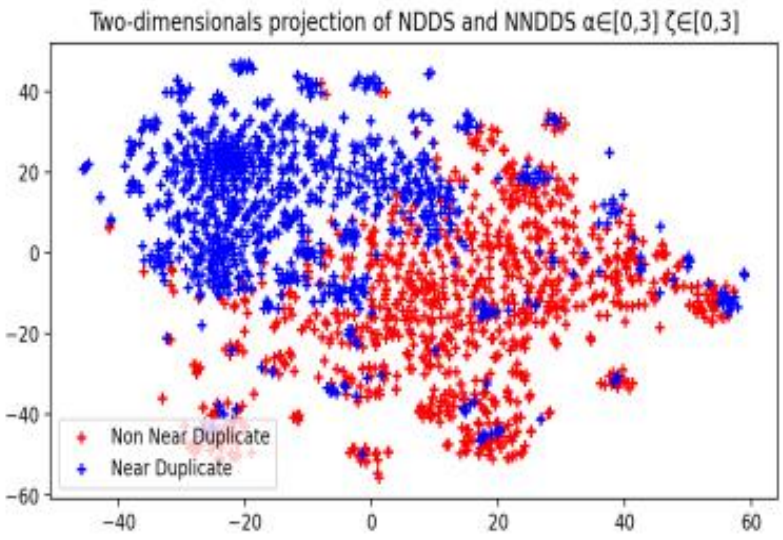
4. Algorithm

The provided algorithms serve crucial roles in the process of generating and validating near-duplicate datasets (NDDS) using machine learning techniques.

Algorithm 1, named RandomDataset, generates a dataset by selecting columns randomly from a given set of columns (universe) and then creating the dataset by randomly selecting elements for each column. This algorithm ensures the creation of datasets with a specified number of columns and lines.

Algorithm 2, AlterateDataset, is responsible for generating a near duplicate version of an existing dataset by adding or deleting columns and lines. It operates by iteratively modifying the dataset based on specified parameters such as the alteration factor (α) and the number of modifications to make.

Algorithm 3, named GenerateExemple, is utilized for generating feature vectors for dataset examples, including both NDDS and non-NDDS instances. These feature vectors are crucial for training and validating machine learning models. By leveraging these algorithms iteratively, researchers can generate examples of NDDS and non-NDDS datasets, extract meaningful features from them, and use those features to train machine learning models to distinguish between the two classes effectively. This process is essential for validating the effectiveness of the proposed method and ensuring its relevance in real-world applications.



5. Experiments

Our experimental setup involves using a Colab notebook with specific hardware specifications, including a Xeon 2.30GHz 4 cores CPU, 25GB of RAM, and a Tesla P100 GPU with 16GB of memory. Throughout our experiments, we employ Algorithm 3 in a loop to generate both learning and test sets. The learning set comprises 100 columns, each containing a minimum of 8000 elements, while the test set consists of 50 columns, each with a minimum of 8000 elements. These datasets are obtained through manual collection from Kaggle dataset 1. The number of lines for each dataset ranges randomly between 50 and 300 in most experiments unless otherwise specified.

In Experiment A, we investigate the influence of the ζ parameter on dataset generation. We generate 7500 examples of each class with various values of ζ (0, 1, 2, 3, or randomly selected between 0 and 4) and train a random forest classifier (200 estimators, max depth 18) to distinguish between NNDS and NDDS. Experiment B explores the impact of the number of columns on dataset generation, varying the number of columns used to build examples and observing its effect on learning and test sets. Experiment C involves evaluating multiple classifiers (adaboost, LGBM, catboost, random forest, TabNet, and stacking algorithm) on learning and test sets generated with different parameters.

In Experiment D, we focus on optimizing the execution time of the algorithm, particularly the feature extraction process, which becomes a bottleneck with large datasets. We propose two alternative methods for feature extraction to reduce computation time: "method 1" involves extracting features from samples of the concatenated dataset and then averaging the results, while "method 2" entails extracting features solely from a sample of the concatenated dataset. These methods are evaluated using datasets randomly chosen between 2000 and 3000 to simulate larger datasets, with experimental parameters consistent with Experiment C. Overall, our experiments aim to analyze various aspects of dataset generation, classifier performance, and optimization techniques to enhance the efficiency and effectiveness of the proposed approach.

6. Results

When considering the influence of the ζ parameter in Experiment A, we observe that the results vary depending on its value. A low ζ leads to inconsistent results due to the algorithm encountering entirely new situations, while higher values of ζ yield more stable results as the system has encountered similar examples during training. Interestingly, our findings suggest that setting ζ to a fixed value, such as 3, produces comparable or even better results than using a random selection for ζ. In Experiment B, which investigates the impact of the number of columns, we discover that the number of columns used in building examples for the learning set does not significantly affect the results. This implies that good results can be achieved without needing to create examples for every possible number of columns. Moving to Experiment C, where we evaluate multiple classifiers, we observe a progressive decline in performance with increasing α and ζ parameters, although the effect of increasing α is relatively small compared to ζ. Notably, the choice between Formula 1 and Formula 2 for feature extraction has minimal influence on most classifiers, except for Random Forest (RF) and TabNet, which exhibit a noticeable loss of accuracy with Formula 2. Despite its specialization in tabular data, the neural network performs below other classifiers, with Catboost demonstrating the best overall results among the evaluated algorithms. Finally, in Experiment D, we analyze the most important characteristics for decision-making in Catboost and RF classifiers, noting the significance of entropy in describing the dataset globally, as well as the presence of distinguishing

statistics on separating characters. Additionally, less frequent characters, such as 'z' and the number 9, are highlighted among the top 10 features, contributing to the classifiers' decision-making process. Overall, these experiments provide valuable insights into the performance and behavior of different parameters and classifiers in identifying near-duplicate datasets.

References

[1] Broder, A. Identifying and filtering near-duplicate documents. Annual Symposium On Combinatorial Pattern Matching. pp. 1-10 (2000)

[2] Alon, N., Matias, Y. & Szegedy, M. The Space Complexity of Approximating the Frequency Moments. Journal Of Computer And System Sciences. 58, 137- 147 (1999)

[3] Van der Maaten, L. & Hinton, G. Visualizing High-Dimensional Data Using t-SNE. Journal Of Machine Learning Research. 9 pp. 2579-2605 (2008)

[4] Panse, F. & Naumann, F. Evaluation of Duplicate Detection Algorithms: From Quality Measures to Test Data Generation. 2021 IEEE 37th International Conference On Data Engineering (ICDE). pp. 2373-2376 (2021)

[5] Herzog, T. Data quality and record linkage techniques. (Springer, 2007)

[6] Papadakis, G., Ioannou, E. & Palpanas, T. Entity resolution: Past, present and yet-to-come: From structured to heterogeneous, to crowd-sourced, to deep learned. (2020), EDBT/ICDT 2020 Joint Conference ; Conference date: 20-03-2020 Through 02-04-2020

[7] Jahns, V. Principles of Data Integration by Anhai Doan, Alon Halevy, Zachary Ives. SIGSOFT Softw. Eng. Notes. 37, 43 (2012,9), https://doi.org/10.1145/2347696.2347721

[8] Ngueilbaye, A., Wang, H., Mahamat, D. & Elgendy, I. SDLER: stacked dedupe learning for entity resolution in big data era. The Journal Of Supercomputing. 77, 10959-10983 (2021,3), https://doi.org/10.1007/s11227-021-03710-x

[9] Wang, J., Shen, H., Song, J. & Ji, J. Hashing for Similarity Search: A Survey. ArXiv. abs/1408.2927 (2014)

[10] Lin, Y., Cai, D. & Li, C. Density Sensitive Hashing. IEEE Transactions On Cybernetics. 44 pp. 1362-1371 (2014)

[11] Draisbach, U., Christen, P. & Naumann, F. Transforming Pairwise Duplicates to Entity Clusters for High-Quality Duplicate Detection. J. Data And Information Quality. 12 (2019,12), https://doi.org/10.1145/3352591

[12] Christen, P. Automatic Record Linkage Using Seeded Nearest Neighbour and Support Vector Machine Classification. Proceedings Of The 14th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining. pp. 151-159 (2008)

[13] Naumann, F. & Herschel, M. An Introduction to Duplicate Detection. Synthesis Lectures On Data Management. 2, 1-87 (2010,1), https://doi.org/10.2200/s00262ed1v01y201003dtm003

[14] Papadakis, G., Skoutas, D., Thanos, E. & Palpanas, T. Blocking and Filtering Techniques for Entity Resolution: A Survey. ACM Comput. Surv.. 53 (2020,3), https://doi.org/10.1145/3377455

[15] Koumarelas, I., Jiang, L. & Naumann, F. Data Preparation for Duplicate Detection. Journal Of Data And Information Quality (JDIQ). 12 pp. 1 - 24 (2020)

[16] Abedjan, Z., Golab, L., Naumann, F. & Papenbrock, T. Data Profiling. Synthesis Lectures On Data Management. 10, 87 (2018,11)

[17] Chevallier, M., Rogovschi, N., Boufares, F., Grozavu, ` N. & Clairmont, C. Seeding Initial Population, in Genetic Algorithm for Features Selection. Advances In Intelligent Systems And Computing. pp. 572-582 (2021), https://doi.org/10.1007/978-3-030-73689-7 55

[18] Andoni, A. & Indyk, P. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. Commun. ACM. 51, 117-122 (2008,1), https://doi.org/10.1145/1327452.1327494