**Chapter 30**

An Effective Duplicate Removal Algorithm for Text Documents

**Amit Jha, Devendra Somwanshi and Mahesh Bundele**

# Introduction

The processes of detection and correction (or removal) of duplicate or inaccurate records from a database are called data cleansing or data cleaning or data scrubbing. There are many ways by which inconsistencies take place; it can be due to corruption in transmission or storage, or may be due to user entry errors, or may be due to different data dictionary definitions of similar entities in different stores. In data cleaning process to remove typographical errors or to validate and correct values through a known list of entities, there are many data cleansing solutions, some clean data by cross-checking with a validated dataset. A number of different activities can be involved in data cleaning such as harmonization of data, standardization of data, and data means, by using standard codes which change a reference dataset to a new standard. The problem of duplicate data removal has been considered a critical aspect of data cleaning. When data from distributed data sources is integrated, the same data in different data sources leads to non-redundant data. This requires the identification and removal of duplicate data that leads to redundant data.

# Data Cleaning Techniques

Based on the existing duplicate records, identification algorithms SNM and MPN

[1] were proposed as an improved algorithm which analyzes attributes and sorts the dataset multiple times to make duplicate records more clustered. The algorithm also gives a special weight to each attribute and introduces the concept of effective weight so that to make the comparison more accurate. A filtering mechanism was introduced

A. Jha D. Somwanshi (B) M. Bundele

Poornima College of Engineering, Jaipur, Rajasthan 302022, India e-mail: [imdev.som@gmail.com](mailto:imdev.som@gmail.com)

· ·

© Springer Nature Singapore Pte Ltd. 2020

G. Mathur et al. (eds.), *International Conference on Artificial Intelligence: Advances and Applications 2019*, Algorithms for Intelligent Systems, <https://doi.org/10.1007/978-981-15-1059-5_30>

269

to improve the efficiency of detection. Many current commercial tools support the extraction, transformation, and loading (ETL) [2, 3] of (possibly unclean) data into a trustworthy (cleansed) database. ProbClean [2] is another solution to efficiently support relational queries. It treats duplicate detection procedures as data processing tasks with uncertain outcomes. There are many defined techniques for data cleaning, some are as follows.

## Parsing

To detect the syntax errors, parsing method is used in data cleansing. By parsing, lexical errors and domain errors can be rectified as it firstly takes sample set of values to deduce the format of the domain. Besides that, for anomaly detection, it generates discrepancy detector.

## Data Transformation

Data transformation is another process of data cleansing in which first of all mapping of data is done from some given format, into a common scheme, which fit it according to the needs, and then it is transformed into the format expected. Standardization and normalization are the part of transformations before the mapping to remove irregularities in data.

## Integrity Constraint Enforcement

Integrity is the major concern when data is modified by inserting, deleting, or updat- ing something. If some integrity constraints are violated, then it is rejected during integrity constraint checking. Additional identified updates are to be added only to the original data if there is no violation in integrity constraint.

## Duplicate Elimination

Duplicate elimination is an essential part of data cleansing. There are a number of methods in duplicate elimination, in every method. In each duplicate detection method, there must be an algorithm which detects the duplication in each entry [4–7].

# Proposed Duplicate Removal Technique

A simple and new approach has been used to design the data cleaning system, especially for duplicate detection and removal which is shown in Fig. 1.

## Duplicate Removal Algorithm

The data from the text file is read word by word and stored in the array list. The length of the array is calculated. In the process, each word/character in the array is compared with all other words/characters to detect the duplicates. If the duplicate is found, it is deleted and the count is maintained for the repeated number of words/characters. The modified array along with the duplicate information is saved.

* Read the data from the file and store it into the ArrayList.
* Calculate the length of ArrayList.
* Initialize variable *i*, *j,* and *c* with 0.
* Check the length of ArrayList
  + If length > *i*, go to step *v,* else
  + store the original ArrayList in file and go to step xi.

*y i* 1

• = +

While (*j* < length-1), go to step vii, else go to step x.

•

Compare value of *a*[*i*] with *a*[*j*], if yes go to step viii, else increment *j* by 1 and go to step vi.

•

Add duplicate element in new ArrayList and Delete duplicate element from the original ArrayList. Count the duplicate element by incrementing the value of *c* by 1.

•

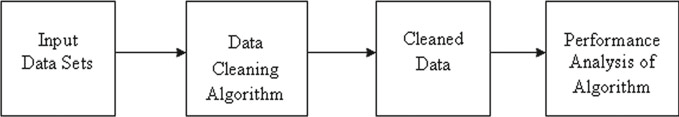
Increment value of *j* by 1.

•

Calculate the new length of updated ArrayList and increment the value of *i* by 1 and goto step iv.

•

* End.



**Fig. 1** Block diagram of the system

# Different Experimental Scenarios

Different experimental scenarios are used to check the performance of our system.

*Text Data:* The data has only text values having some duplicate text data.

*Numeric Data*: The input data consists of numerical values with some duplicate numerical values.

*Text and Numeric Data*: The data is the combination of text and numerical values.

*Text, Numeric, and Special characters*: In this data file, there are text data, numeric data, and some special characters with duplicate data also.

*For testing* the algorithm for larger datasets, files with varying numbers of words from 10 to 23,598 including all data types were used.

# Experimental Results

Our experiments were performed on jdk 1.5.0 installed on 2.1 GHz Intel Core i5 processor and Windows 7 Operating System.

## Text Data

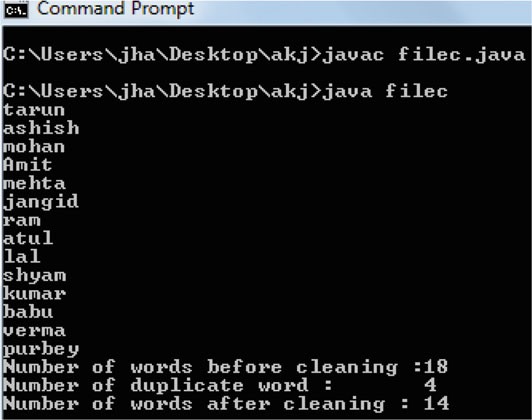
For testing the algorithm with text data, following 18 text words with four duplicates were tested and the screenshot displayed i.e. Fig. 2 shows that all the four duplicates are removed from original data.

Tarun ashish mohan tarun Amit Mehta Jangid ram atul Amit Lal shyam mohan kumar Babu verma purbey Atul

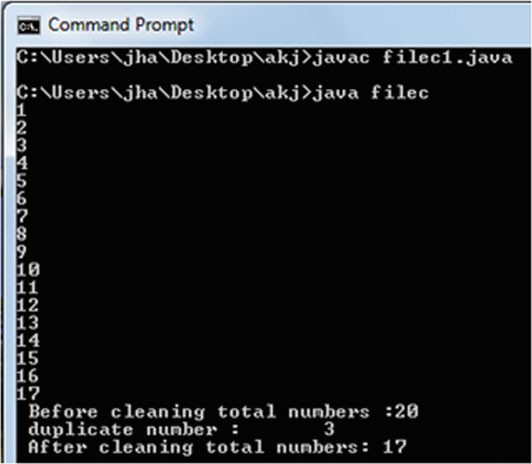
The total number of words before cleaning was 18 in which four words were dupli- cate ones. After applying our method, the result shows excellent result by removing duplicate words in the data.

## Numeric Data

For testing the algorithm with numerical data, the string of following 20 numeric values with three duplicates was fed as input to the algorithm and the results are shown in Fig. 3.



**Fig. 2** Performance analysis on text data



**Fig. 3** Performance analysis on numeric data

1 2 3 4 5 6 7 8 9 10 11 12 13 14

15 16 17 1 2 3

Before cleaning total numbers: 20 Duplicate numbers: 03

After cleaning total numbers: 17

It shows that the algorithm works good with numerical data also.

## Text and Numeric Data

For testing the algorithm with text and numeric values, a file containing 55 total numbers of text words and the numerical values was tested.

Ram Sumit Ram 5 16 Shyam Dheeraj Amit 6 17

Sita Sanjay Alap 7 18 Gita Garima Sumit 8 19 Manoj

Raj Dheeraj 9 20 Ravi Rajnish Garima 10 21 Amit Surendra

Manoj 11 22 Alap Shusil 1 12 23 Amit Shashi 2 13 24 Amit

Shyam 3 14 25 Amit Sita 4 15 26

In Fig. 4, results show that all the duplicates are removed and the system is performing well in this case.

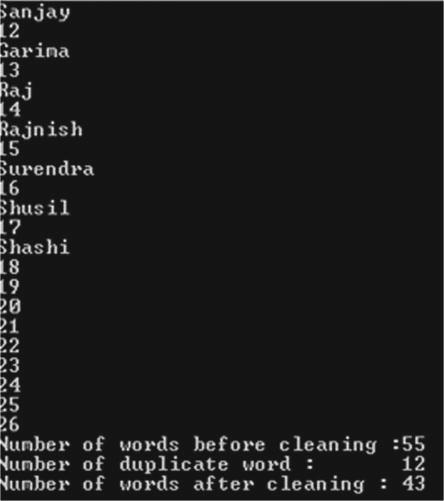
Number of words and number before cleaning: 55 Numbers of duplicate word and number: 12 Number of words and number after cleaning: 43.

## Text, Numeric, and Special Character Data

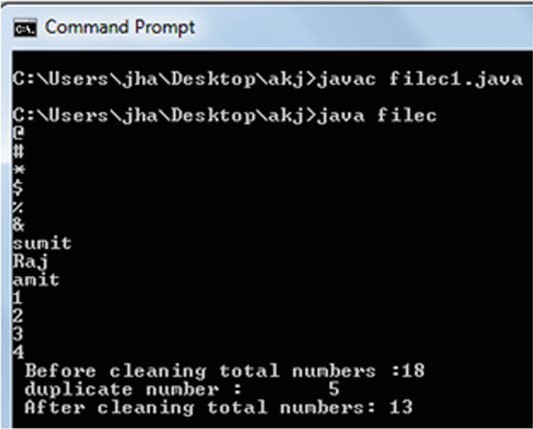
A file containing total 18 characters including text, numeric values, and special symbols was fed to the algorithm, and the test results show that five duplicates were removed which is shown in Fig. 5.

@ @ 1 # & 2 \* sumit 3 $ raj 4 % Amit 1 & raj 2 Number of words, Number and Symbols before cleaning: 18 Numbers of duplicate: 05

Number of Data after cleaning: 13



**Fig. 4** Performance analysis on text and numeric data



**Fig. 5** Performance analysis on text, numeric, and special character data

**Table 1** Performance analysis on varying numbers of words

|  |  |  |  |
| --- | --- | --- | --- |
| File size | Number of duplicate words | Number of words after removing duplicates | Duplicate detection rate (%) |
| 10 | 03 | 07 | 100 |
| 25 | 07 | 18 | 100 |
| 65 | 13 | 52 | 100 |
| 95 | 27 | 68 | 100 |
| 133 | 35 | 98 | 100 |
| 356 | 148 | 208 | 100 |
| 578 | 189 | 389 | 100 |
| 753 | 256 | 497 | 100 |
| 1243 | 365 | 878 | 100 |
| 1986 | 487 | 1499 | 100 |
| 2576 | 583 | 1993 | 100 |
| 3563 | 654 | 2909 | 100 |
| 4587 | 666 | 3921 | 100 |
| 5643 | 743 | 4900 | 100 |
| 6487 | 798 | 5689 | 100 |
| 7489 | 1008 | 6481 | 100 |
| 8734 | 1154 | 7580 | 100 |
| 9679 | 1678 | 8001 | 100 |
| 18,765 | 4563 | 14,202 | 100 |
| 23,598 | 5003 | 18,595 | 100 |

## Performance Analysis on Varying Numbers of Total Words

To test the algorithm on a file that contained huge number of data, different size files are tested. The results are excellent as expected and shown in Table 1.

# Conclusion

A data cleaning algorithm is presented in order to detect and remove duplicate data from text files having different data types in order to improve the quality of data. From the results, it could be found that the proposed algorithm could provide 100% correct duplicate detection rate in all the cases. The proposed concept can be extended for removal of other kinds of data impurities. This work presents testing with text files only, and hence, the algorithm can be modified for duplicate removal from other file formats such as xls, xlsx, doc, docx, and pdf.

# References

1. He L, Zhang Z, Tan Y, Liao M (2011) An efficient data cleaning algorithm based on attributes selection. In: 2011 IEEE 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), pp 375–379
2. Beskales G, Soliman MA, Ilyas IF, Ben-David S, Kim Y (2010) ProbClean: a probabilistic duplicate detection system. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE), pp 1193–1196
3. Maddodi S, Attigeri GV, Karunakar AK (2010) Data deduplication techniques and analysis. In: 2010 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET), pp 664–668
4. Ali K, Warraich MA (2010) A framework to implement data cleaning in enterprise data ware- house for robust data quality. In: 2010 IEEE International Conference on Information and Emerging Technologies (ICIET), pp 1–6
5. Higazy A, El Tobely T, Yousef AH, Sarhan A (2013) Web-based Arabic/English duplicate record detection with nested blocking technique. In: 2013 8th International Conference on Computer Engineering & Systems (ICCES), pp 313–318
6. Zhang J (2010) An efficient and effective duplication detection method in large database appli- cations. In: 2010 IEEE 4th International Conference on Network and System Security (NSS), pp 494–501
7. Lwin T, Nyunt TTS (2010) An efficient duplicate detection system for XML documents. In: 2010 second IEEE International Conference on Computer Engineering and Applications (ICCEA), vol 2, pp 178–182

[View publication stats](https://www.researchgate.net/publication/339561834)