PRAGMATIC AND SCALABLE DEDUPLICATION TACTICS USING HEURISTIC PRUNING ALGORITHM

Kari Venkatram

Research Scholar, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore – 632 014, Tamil Nadu, India,
venkat_kari@yahoo.com

Geetha Mary A

Associate Professor, School of Computer Science and Engineering, Vellore Institute of Technology, Vellore – 632 014, Tamil Nadu, India,
geethamary.a@gmail.com
*Corresponding Author: Geetha Mary A
Fax: (+91) 416-224-3092

Abstract
Master data acquired from many legacy systems and heterogeneous sources, hence it's possible to get duplicated and it influences the quality of the master data services such as all CRUD operations & other business services. The data deduplication process is one of the most important requirements for building comprehensive master data management (MDM). There are several custom and hybrid models available for data quality improvement, but can't resolve all the quality issues such as data duplication. This article portraits, how duplication happens, and what are the measures to prevent duplicate data. The proposed solution - pragmatic and scalable approach to identify and eliminate the duplicate data from the repository. The solution is based on a heuristic pruning algorithm with a cosine similarity measure for eliminating data duplication. This algorithm yields better time O ($2mn - m^2$) complexity compared to the normal cosine algorithm. This approach is built with a two-step process, the first step is the deduplication process of existing data, and the second step is to prevent the duplicate data being entered into the real-time system. It is been observed with the proposed solution that, the quality of data is improved close to 90% as part of the deduplication process.

***Keywords*- Deduplication, cosine similarity, heuristic pruning algorithm, data quality, master data management**

1. Introduction to data quality in Master Data

Master Data Management (MDM) is a comprehensive framework to create a master file with records of master items of an enterprise as a common point of reference. This file describes entities of business such as customers, distributors, products, locations, partners and employees, etc., MDM is critical for an enterprise/ organization to link all the referential data with each other into one file, called a master item[1]. This framework enables streamlining of data sharing effectively, uniformly, and consistently among master items [2] within MDM.

The data quality in MDM is important to make the right decisions and achieve a competitive advantage as compared to the competitors. Poor quality of data will result in bad decisions [1]. The data quality issues arise from the data being 'silos' due to factors such as consistency, accuracy, completeness, timeliness, and relevancy, etc., [2]. If data quality is poor then it is practically useless and sometimes even dangerous. All the departments will suffer to produce accurate reports and predictions due to poor quality data. For instance, the Sales and Marketing department may lead to inefficient conversion of sales, the customer services department may maintain poor customer relationships and may churn out customers and damage the reputation of the organization. Quality issues lead to excessive usage of personnel and resources for inflexible or unresponsive customers. Business operations department will suffer with overloading data clean up and distraction of the compliance and associated challenges.

This article aims to build the deduplication strategy for master data (depicted the process with Customer data) efficiently. Lack of an explicit data deduplication strategy in Customer Relationship Management (CRM) systems resulting in duplicate of customer data. There are many reasons for data being duplicated such as human errors by not searching for customer data before adding customer records (Name, location, accounts, deals, or leads), alerts

by the system are ignored by the person who is trying to insert data. The tendency is more for duplication of customer data due to digitization of businesses and as it will be using various devices to access their data.

Another source of duplication is, data imported from various systems using different tools. The tool may not be efficient to eliminate duplicate records. Though the duplicate records are flagged as "potentially duplicate" while importing the data, users tend to ignore the alert and create a new record.

Data integration among the ecosystem such as third party data providers, external data sources - website registration forms, partner portals, message brokers, or other applications may not validate the duplication of records against CRM. At times, external data may not be able to update CRM; this can cause new records in CRM even though it is identified as duplicate.

Due to clerical issues, application limitations and issues in software tend to create duplicate records in the system. Higher quality data can be achieved with lower duplicates in the system and similarly higher the duplicates the lower data quality, also credibility of the reports will be less and business decisions will be affected due to poor quality. Duplicates above 15 percent of data is a very serious concern and can threaten the business.

2. Challenges and Scenarios for data duplication

At times duplication of data may seem not important than the rest of other issues. Duplicate data can cause very serious consequences such as erroneous reports resulting in bad decisions and very expensive steps to correct it. Usually, data duplication happens in business across different departments due to the use of different software applications. For instance, customer and product data are often entered repeatedly into accounting, inventory, product management, customer relationship management software applications. The same data can be entered into different business applications of the enterprise causing duplicate information of the same customer. Invoicing and dispatch of products can go wrong with duplicate records. This can be highly expensive. The financial analysis also can go wrong. For example, promotional mailers may be sent multiple times to the same customer increasing postage/printing expenses, which can be very much avoided. Unnecessary data will be stored in systems. As per Gartner's impact analysis on bad data management close to 25% revenue dip[3] happening due to duplicate data. A single source of truth (SSOT) is not possible due to data duplication. The profiling of the customers and analysing customer behaviour is not accurate with duplicate data. For instance, no transactions of one particular customer are considered then the ability to market the product/services is reduced and consequently, the customer care team's service will be inefficient. Different data duplication scenarios are discussed in the subsequent sections.

Data duplication scenarios: This section describes major data duplication scenarios with an example of customer data. The scenarios covered in this section are approximately 90% of total customer data duplication. If these scenarios are addressed, the quality of the customer data will improve close to 90% and produce good results.

Name standardization: Customer data quality gets into a serious issue due to non- standardization of customer names. Nearly 5% of data duplication of customer names can happen if the system is not standardized. Customer names are accepted with all the alphanumeric, junk, special characters, etc. leading to confusion and creation of duplicate records. As shown in Table 1, a customer name Five Star technologies can be represented as many means. This name should be standardized by avoiding special characters and numbers in the attribute name.

#	**Customer Name**
1	Five Star Technologies
2	5 Star Technologies
3	5* Technologies
4	***** Technologies

Table 1: Customer Names without standardization

Master data ecosystem can be vulnerable to get duplicated due to the usage of certain abbreviations or with the usage of jumbled names. Here is the example of an organization name "Tech Mahindra Limited" in Table 2 are represented as given below.

#	**Customer Name**
1	Tech Mahindra Ltd
2	Mahindra Tech Ltd
3	TechM Ltd
4	Tech Mahindra Limited

Table 2: Customer Names with jumbling words and abbreviations

Names with additional spaces or missing spaces are also a very common scenario, which leads to duplication. As shown in Table 3, the company name “Tech Mahindra Ltd” is depicted differently with leading or trailing spaces.

#	Customer Name
1	Tech Mahindra Ltd
2	Tech Mahindra Ltd
3	Tech Mahindra Ltd
4	Tech Mahindra Ltd

Table 3: Customer name with some unwanted spaces

The system should eliminate the extra spaces between the words and should be able to manage the jumbling of the words. Keying in customer addresses differently for different instances of the same customer also can lead to duplication. As depicted in Table 4, customer addresses are spread across four attributes differently.

#	Name	Add1	Add2	Add3	Add4
1	AT&T Istel	227	Marsh wall	Sovereign house	Polar
2	AT&T Istel	227 Marsh wall	Sovereign House, Polar		
3	AT&T Istel	227 Marsh wall, Sovereign house	Polar		
4	AT&T Istel	227 Sovereign house	Marsh wall	Polar	
5	AT&T Istel	227 Marsh wall	Sovereign house	Polar	

Table 4: Customer address with multiple data fields

These types of issues arise due to not following standard conventions across departments/applications of an organization. If geographic elements such as city, county, state, province, zip code, etc., are not standardized, lead to duplication of customer data. Here is an example is shown in Table 5.. Zimbabwe has no postal code system in place and hence the postal code is not mandatory required. However, some of the systems mandate the postal code. Users may enter some junk data, as it is mandatory in their application leads to duplicate records.

#	Name	City	State/ Province	Country	Zip Code
1	The Wattle Company	Mutare	Manicaland	Zimbabwe	
2	The Wattle Company	Mutare	Manicaland	Zimbabwe	000263
3	The Wattle Company	Mutare	Manicaland	Zimbabwe	263000

Table 5: Customer data without postal codes

Some countries like Singapore may not have a state or province in their address. As shown in Table 6, sometimes users tend to fill some junk value or some details of other fields in the state/ province column. This will lead to the creation of a duplicate record.

#	Name	City	County	State/ Province	Country	Zip Code
1	Arinso Singapore Pte Ltd	Singapore			Singapore	239920
2	Arinso Singapore Pte Ltd	Singapore		UE Square	Singapore	239920
3	Arinso Singapore Pte Ltd	Singapore	Clemenceau avenue		Singapore	239920

Table 6: Customer data without the State of Province

Wrong information such as agent information or user-specific information is fed into some irrelevant attribute. This will cause wrong/duplicate records as shown in table 7.

#	Customer Name	Address1	Address2	Address3	Address4
1	A T &T ISTEL	227	MARSH WALL	SOVEREIGN HOUSE	POLAR
2	A T &T ISTEL	PO 123458382878	227 MARSH WALL	SOVEREIGN HOUSE	POLAR
3	A T &T ISTEL	123458382878	227 MARSH WALL	SOVEREIGN HOUSE	POLAR

Table 7: Customer data with some additional information in address fields

There is another scenario where users can have multiple addresses with the same physical location. For example, one customer can have two or three sites each site in a different building in the same location. Tech Mahindra might have one physical location in Bahadurpally, Hyderabad and it might have two different buildings like IT-1 and IT-2. From each building, there can be different orders for goods and their delivery addresses are different. However, the customer remains the same. This is another scenario, which can create duplicate records.

One of the usual scenarios is the usage of phonetic names. This can create confusion while recording customer data and prone to duplication. Example - “Reliance Trends”. This can be written as “Reliance Trendz” by one user and can be written as “Reliance Trends” by another user. This will create a duplicate record.

3. Literature review and deduplication strategies

Entity duplication is a very common and well-established problem in the industry especially textual corpora[4]. Suppose there are N entities such that, set of entities $X = \{x_i | x_i \in R\}$ for i from 1 to N. R is the complete set of data. Data duplication can be defined as below: Let consider the data set X, with an entity q in it, if the system tries to find its closest entity in X or closest entities (subset $X' \subset X$) where the distance from the entity in X' to the entity q is less than the threshold value. It approximates similarities among the entities[5] q and X'.

There are several proposals by researchers for efficient similarity detections the important ones are discussed in this section. Charikar proposed an algorithm based on hash functions[6] called SimHash (SH) which uses random projections as a set of hash functions and creates an angle between two entities. This will work majorly for linearly separable data. Further, it was enhanced by Kulis et all for catering non linearly distributed data as well, this is called Kernelized Locality Sensitive Hashing[7]. Then Wang et all have enhanced it with an improvement in hashing which has its prior knowledge. This is called Semi-Supervised Hashing (SSH)[8]. Many others also proposed several studies on this through locality-sensitive hashing, spectral hashing, etc.[9].

Though there are several hashing functions for identifying text similarities, it is a unique problem to identify the similarities of business entities. These unique problems in entity similarities require a very specific customized algorithm to address such data duplication.

Section 3 discussed various scenarios of how and why the customer data is duplicated and persist in databases. There are some existing solutions and strategies to deal with this. This process is generally called a deduplication process. Data deduplication is a process of eliminating duplicate data and restricting the data being duplicated. This process enables a preventive and corrective action also called data cleansing. Deduplication is used to identify duplicate records that refer to the same real-world entity and help us to merge the duplicate records through data linkages, record matching techniques[10]. In this process, all the relevant, accurate data will be identified by eradicating duplicate records.

As discussed in section 3 on the various data duplication scenarios, it is critical to identify the right strategy for data deduplication. The deduplication strategies would be applied at various levels at OLTP and OLAP. The deduplication strategy should be considered on high priority, as it is a better option to reduce further duplication of the data in the ecosystem.

There are several cleansing techniques/tools are available to integrate within the OLTP/source system. If applied, cleansing will happen at the source system and eliminate the duplicate data within the source system/initial layer, hence data duplication is avoided.

Since the data at the source system is being cleansed, the quality of data is better. Organizations invest cleansing techniques at the source layer rather in downstream systems, as it is a better option. Source system cleansing has various benefits as downstream systems benefit from high-quality data. In case the deduplication strategy was done at the warehouse level or data lake level, the source databases or source systems will still be having duplicate records, eventually, all the downstream applications depending upon the source data/ source system will require to cleanse the data separately for better data quality. Though it is recommended to do the deduplication at the source level, there are certain challenges in implementing it at the source level, especially for high performance/real-time transactional applications. The actual challenge here is the real-time systems require an immediate response from the services where the deduplication service may not meet the required SLA, due to various business rules. In such cases, deduplication is tough at source level and it is advised to do it in the downstream systems such as data warehouses or data lakes.

Deduplication processes are applied to different platforms and different domains across the enterprise areas to ensure the businesses get the best out of their data. It must be applied to SCM (Supply Chain Management), ERP (Enterprise Resource Planning), BI (Business Intelligence), and Customer Relationship Management (CRM) databases. SCM can support not only their data but also the network of supply chain business processes with reliable, practically applicable data. ERP is data-centric and normally designed around key business processes like financial management, materials management, human resources management, and inventory control management, etc. Hence, data quality plays a vital role in this area as well. BI- is the backbones of DSS (Decision Support Systems) of an enterprise. Hence, deduplication essential for achieving the goals of DSS. CRM has customer data, which is very sensitive and holds key information about customer preferences and their attributes. To build customer relationships strong and having a good partnership with customers, organizations must enhance the quality of customer data. The key to more efficient CRM is a unique view of the customer across the enterprise.

Above mentioned solutions and strategies are not comprehensive for the deduplication process across the enterprise, as it is applied at each business process level, hence it is not sufficient solution to address the duplication. So it is required to have a comprehensive solution to deal deduplication at a common level across the business processes.

There are several proposals from researchers on the exploration of vector space model, engram, and similarity functions using the weight of the words with inverse document factory[11] (IDF). These approaches predominantly used for the deduplication of data with complex structures. There are several other approaches for evaluating the similarity scores/ranks for identifying duplicate records. The similarity of records is usually evaluated through vector space models[12] where the large fragmented text is constructed from different fields and calculate the similarity among the records using edit distance measure such as Levenshtein distance[13] (LD). The number of deletions, insertions, calculates distance, or substitutions required to transform Source String(S) into Target String (T). The distance between source and target represents the way the two strings are similar or dissimilar. In other words, shorter distance represents more similarity and vice versa.

Another measure is Jaccard distance, which measures dissimilarity between source and target strings. This method is used to compare and find the proximity of the data among the data clusters[14]. Machine learning techniques are also used to identify similar records and eliminate them by Atlas systems, which performs mapping between records to integrate different data sources. Other approaches to document clustering, agglomerative hierarchical clustering, and k-means clustering to identify the similarities among the data. [15]

All measures and articles, which were discussed above, are very generic solutions for identifying the similarity or dissimilarity in due course of the deduplication process. However these are not specific solutions designed for deduplication of an entity of master data, hence it is an essential need for a robust and scalable algorithm/ framework to address the entity deduplication of master data.

4. Proposed solutions for deduplication & strategies

As discussed in the previous sections there are a lot of challenges in identifying duplicates in master data. For instance, a customer entity is built up with multiple text fields that are rapidly generated from different source systems and it can be achieved with text similarities/rank algorithms. The similarity coefficient represents the similarity between two documents or two strings or queries. There are few similarity coefficients such as the Dice coefficient, Jaccard coefficient, Pearson correlation coefficient, and cosine coefficient[16, 17]. The cosine similarity coefficient gives better results compared to others[16].

For attribute similarity evaluated several similarity measures below are the empirical analysis of the similarity measurement. Considered the following Levenshtein (LEV), Longest Common Subsequence (LCS), Jaro-Winkler (JWC), and Cosine Similarity (CS) in this study. Compared two entities with 6000 variations how different similarity algorithms perform concerning the accuracy, sensitivity, specificity, precision, and the false alarm was recorded. In our study, we conclude using the Cosine Similarity model for similarity calculations as it outperforms as compared to all other measures.

Algorithm	TN	FP	FN	TP	Accuracy
LEV	701	1	656	4642	0.8905
LCS	700	2	312	4986	0.9477
CS	695	7	140	5158	0.9755
JWC	701	1	178	5020	0.9697

Table 8: Attribute Similarity Evaluation for Entity A (TP: 5298, TN: 702)

Data clustering is a commonly used technique that helps to organize the dataset by grouping them into semantically consistent buckets called clusters. It is achieved by identifying the similarities based on the similarity coefficient[18].

As depicted in Fig. 1, customer data (text data of name and address) will be fed into the system to create a hierarchal cluster. The preprocessing is an important step of the algorithm, which removes stop words, delimiters from the text data, and fed into the next phase. In this step, each word in the string – called the token will be passed through the synonyms repository. If any synonym is found, word will be replaced in preprocessing text as a token. This will continue to identify the synonyms based on the phonetics of sound using Soundex functions.

The next phase is data bags, in this phase, each word in the record will be calculated and stored in data bags for further processing. The data will be stored as a table with the occurrence frequency of each word in the record. The next phase is the calculating distance between records with cosine similarity distance measure and identifies

the distance or score between two different records. Similar records grouped under one cluster. Duplicate records are grouped into one cluster and hence duplicates can be eliminated by liking or removed.

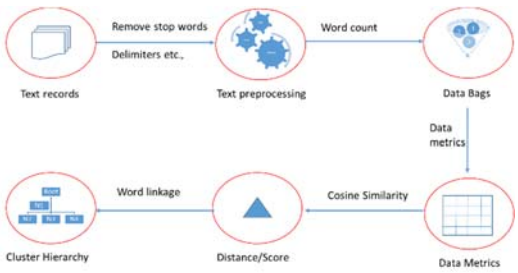

Fig. 1. Clustering data flow

As depicted in Fig. 2 - deduplication architecture, various upstream systems such as SCM, ERP, CRM, Internet content, etc. are the source systems to create customer data. All these customer records being created from various systems are pushed to a single source of truth (SSOT) called master data (Customer Master). Distributed messaging system (DMS such as Kafka) is a message broker of the ecosystem to push the data from various sources to MDM; simply DMS takes customer data from different producers and shares the messages to different consumers. Before giving it to the end consumers, the messages will be proposing a middle layer-distributed in-memory database (DIMD such as Apache Spark) help in processing the data at lightning speed. The processed data will be pushed/flown to other downstream databases or the distributed search engine (DSE such as Solr). The end-users consume the services from the downstream database or distributed search engine. All duplicate data will be eliminated in this distributed middle layer.

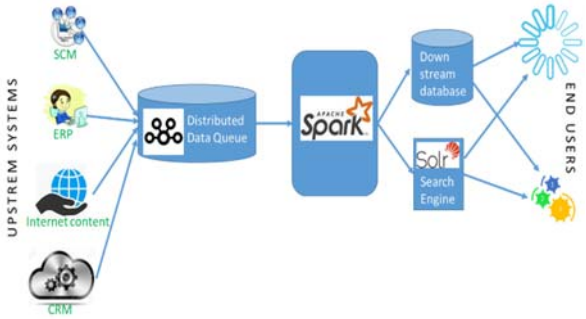

Fig. 2. Deduplication architecture

As shown in the architecture above the deduplication process happens in the distributed in-memory database (middle layer) of the architecture. Deduplication is achieved here with a two-step process. The first step is to one-time process to eliminate all the duplicates in the existing data.

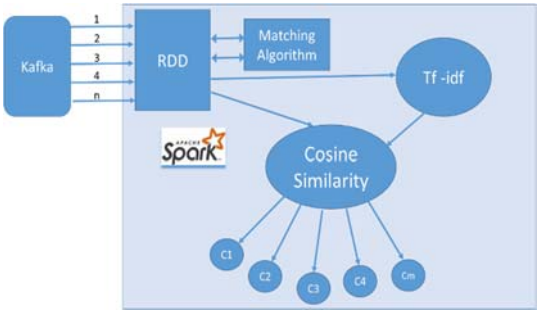

Fig. 3. One Time process for Data deduplication

As depicted in Fig. 3, all duplicate data in the existing system will be eliminated with a one-time job. A matching algorithm based on cosine similarity measures will match the duplicate entities as described in section

3. Heuristic pruning algorithm with Cosine similarity is used to cluster similar records with similarities over 90%. All the records (let say n) are loaded into Spark RDD/datastore after clustering into a set of unique groups. The unique groups (say m) will be less than or equal to n. Such clusters are used to link the related entities into one single customer hierarchy. The cosine similarity algorithm is built based on two significant terms such as Term Frequency (TF) and Inverse Document Frequency (IDF). Term Frequency also is known as TF, measures the number of times a term (word) occurs in a document. In any large document, the frequency of the terms will be much higher than what we usually think of. Hence, we need to normalize the document based on its size. The simplest way is to divide the term frequency by the total number of terms. For instance in a document, if the term "tech" term occurs ten times. The total number of terms in the document is 100. Then the normalized term frequency is 10 / 100 = 0.1. This is the way; the normalized term frequency is calculated. In some cases, normalized term frequency is calculated with log frequency weights as shown below:

$$\sum_{t \varepsilon q \cap d}(1 + \log tf_{t,d}) \quad (1)$$

Document frequency (df of term t) is the number of documents that contain term t. The inverse document frequency is used to measure document frequency across documents for the given term. To reduce the impact of common term vs rare terms across all documents, inverse document frequency can be used to normalize the impact. The logarithmically scaled inverse fraction of the documents that contain the word is obtained by dividing the total number of documents (N) by the number of documents containing the term (df_t), additionally applying the logarithm to the quotient. Therefore, certain terms that occur too frequently have little power in determining the relevance, it gives a way to weigh down the effects of too frequently occurring terms. Besides, the terms that occur less in the document can be more relevant, it gives a way to weigh up the effects of less frequently occurring terms. Hence, logarithm helps in solving the problem of high and low frequent terms. So df_t is an inverse measure of the informativeness of t, hence df_t must be less than or equal to no of documents N.

We can define the IDF (inverse document frequency) of term t as below[19] :

$$idf_t = log_{10}(N/df_t) \quad (2)$$

$log_{10}(N/df_t)$ used instead of N/df_t to reduce the effect of high frequency occurring terms which are not more relevant than low frequency occurring more relevant terms. This is relevant for two are more terms; however, the effect of idf on ranking for one-term queries is nothing. In this article only two documents are compared at any single point of time, hence idf has defaulted to 1. However, it is to be considered with real-time search scenarios.

Tf-idf weighting: The tf-idf weight of a term is the product of its tf weight and its idf weight. To build the cosine similarity equation, it is required to solve the equation of the dot product of two vectors where it will measure the similarities between two documents. The Algorithm is defined as below: Every term in the document tf-idf is calculated. For the same words in other documents, calculate tf-idf for each word. There will be two vectors, i.e., the first vector is tf-idf weightage for the first document, and the second vector is tf-idf weightage for the next document. Now calculate the dot product of these two vectors, which is a scalar value. This scalar value is the relevancy or similarity of these two documents. The cosine similarity between two vectors (or two documents on the vector space) is a measure that calculates the cosine angle between them[20]. This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space because the magnitude of each word of each document and the angle between the documents were considered of each count (tf-idf) of each document and the angle between the documents[21]. Cosine Similarity generates a metric to measure the similarity between two documents by looking at the angle instead of magnitude. If the cosine similarity for two different documents is "1" means that both the documents are in the same direction and the angle between them is 0 degrees whereas cosine similarity value is "-1", then both the documents are in opposite direction and their angle is 90 degrees.

Note that even if we had a vector pointing to a point far from another vector, they still could have a small angle and that is the central point on the use of cosine similarity, the measurement tends to ignore the higher term count on documents. Suppose we have a document with the word "sky" appearing 200 times and another document with the word "sky" appearing 50 times, the Euclidean distance between them will be higher but the angle will still be small because they are pointing to the same direction, this matters a lot while comparing the entities hence cosine similarity has been considered for this algorithm.

Proposed heuristic pruning (HP) algorithm to cluster similar records into one group with low running time compared to regular cosine algorithm:

Algorithm: Heuristic Pruning (HP) algorithm with the cosine similarity measure
Input a set of entities with n attributes (E)

Output a set of entities with cluster id, overall score, attribute group-wise scores in it (E')

Comments Entity (E) is an input object with several attributes in it. Example E ($a_1, a_2, a_3..a_n$) where E is an Entity may be customer a_1 : customer id, a_2: name, a_3: add1, a_4: add2,.., …a_8: state, a_9: country, a_{10}: pin

Require set of E = {E_1,E_2, E_3..$E_{|E|}$}

1. *Start* HP Algorithm
2. {E} <- fetch entities from DB
3. ThresholdValue <- 0.90
4. startRecord<-1
5. PartyID <-0
6. *For each* E in {E}
7. entity <- dataList.next()
8. *If* entity.a11 is not null
9. dataList.remove()
10. Repeat
11. *Else*
12. custData <- Concatenate all the columns ($a_1, a_2, a_3..a_n$)
13. splitCustData <- split custData
14. compareUniqCustData <- unique words from splitCustData
15. a_1 <- entity. a_1
16. a_n <- entity. a_n
17. entity. a_{12} <- 1
18. *For* initial<- startRecord+1, dataList.size()
19. nextEntity <- dataList.next()
20. nextCustData <- Concatinate all the columns ($a_1, a_2, a_3..a_n$)
21. calculate the normalized term the frequency for all the words in compareUniqCustData vs custData
22. fromCustVector <- Generate term frequency vector
23. calculate the normalized term the frequency for all the words in compareUniqCustData vs nextCustData
24. toCustVector <- Generate term frequency vector
25. score <- calculate the dot product of fromCustVector X toCustVector
26. *If* score > ThreshooldValue
27. entity. a_{n+1}) <- a_{n+1})
28. entity. a_{n+2}) <- a_{n+2})
29. *Repeat*
30. *End If*
31. startRecord <- startRecord+1
32. *End For*
33. *End If*
34. *End For*

Since this algorithm built based on the heuristic pruning algorithm, the running time of the algorithm is considerably less compared to the regular algorithm. The algorithm for next iteration prunes the records, which are already clustered, hence it reduces the number of records for next iteration onwards.

Time complexity analysis of heuristic pruning (HP) algorithm can perform better than conventional algorithms. Initialization at the beginning of the algorithm can be done with a constant no of primitive operation per element and it takes O(n) times. There are two nested loops, which are controlled by start record, and the entire loop with a condition to resume the loop. The body of the outer loop would contribute the number of primitive

operations proportional to n and m, hence it takes O(n-m) times. The body of the inner loop is also controlled by start record which is incremented and controlled, which implies the statements in the inner loop contributes to $O((\mathrm{n}-\mathrm{m})\ 2)$. times

So total running time of the algorithm is given by the sum of all the above terms asymptotically would be: $O((n-m)^2)$.

Here n is the total number of entities and m is the total number of duplicate entities in the set of n entities. A regular algorithm running time asymptotically would be O(n^2), so the running time for the proposed algorithm is improved by $2mn - m^2$. After a time setup is done all the existing duplicate records are eliminated in the system; however in the online process if any new record is getting created or modified will be verified against the processed de-duplicated data with a similar algorithm to avoid duplications.

Online Deduplication: As discussed earlier sections, the second step is to avoid generating duplicate entities when a new entity is being created or modified online. In this process, the same matching algorithm is used in conjunction with a search engine used for online deduplication. As depicted in Fig. 4, the source system's user will search before creating a new or modifying the entity. If there is any entity matching the input, the search will fetch the data from the search engine based on the matching algorithm with the relevancy or score of the record. The highest matched record that scores over 90% will be returned if it is available otherwise a new record is created. Hence, it restricts the user in creating duplicate records from the online process.

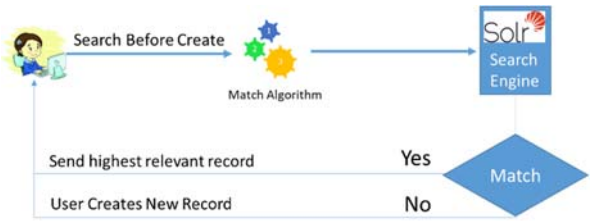

Fig. 4. Online Deduplication

5. Metrics and Analysis of proposed solution

Experiment Settings: Master data samples used in this experiment are customer data of an industry leader. Customer data is taken such as customer name, address, and geographical elements. The data used in this experiment is the fair distribution from the entire data set, which is used for similarity function.

The proposed solution is validated against the test dataset. We observed and assessed the model carefully for inferring the model quality. Evaluated the performance of the proposed solution with a good number of data with an adequate possible duplication of data within. Then ran this data with this proposed solution to identify the duplicate records based on heuristic pruning cosine similarity algorithm and observed that total records are consolidated into multiple groups/ clusters. Upon manual verification of thousands of records, we found that the clusters with cosine similarity accuracy are very high in this case and it is above 90%. Few scenarios where some of the special cases like post box numbers are given as NW-North West, SW – South West, etc., were not matched, however, these also can be addressed by improving the existing logic.

As depicted in the following in Fig. 5, the customer entities are scattered across space with the original dataset and were distributed across space with lots of duplicates in it. There are 5000 entities are distributed to close to 5000 clusters.

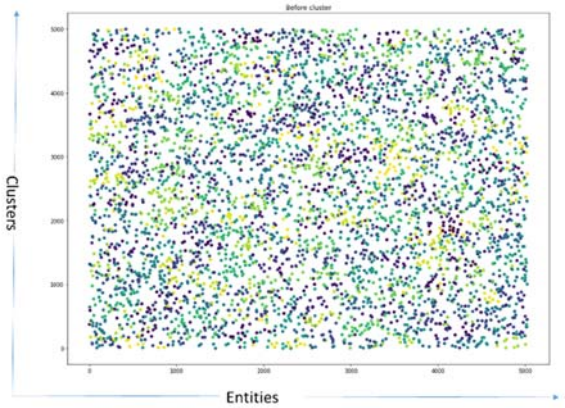

Fig. 5. Original Customer data with existing clusters

With the proposed HP algorithm, ran a one-time deduplication process on the original dataset and observed that quite a few records were consolidated into one group. As depicted in Fig. 6, after running the dedupe process observed that all the data points/ customers belong to one group are separable from other clusters. From this picture observed all 5000 entities are consolidated to 600 clusters. All similar entities are grouped and consolidated.

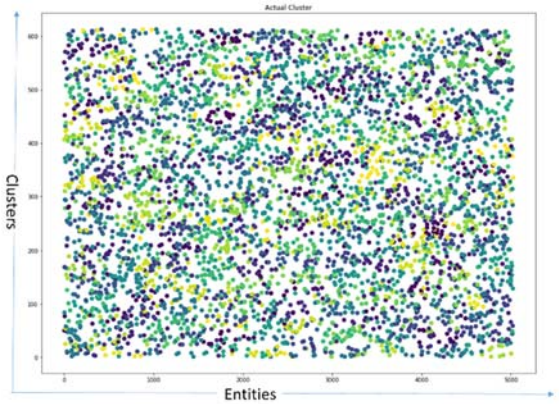

Fig. 6. Customer data after running the dedupe process

Data points after the deduplication process are depicted in the following Fig. 7 Observe the data points in the plot many of the data points are clustered together. It indicates that the records with high similarities are consolidated and consolidated records are duplicate records which can be eliminated or generate a hieratical entity.

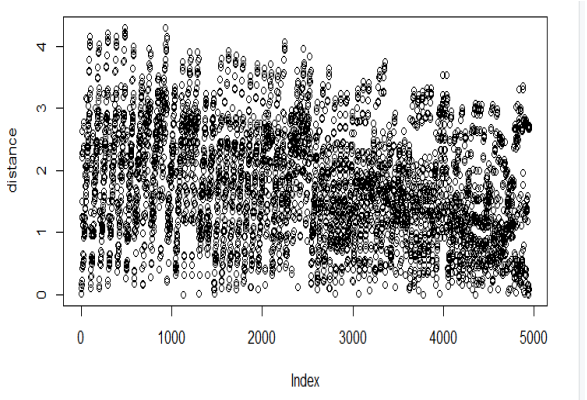

Fig. 7. Deduplicated data for a huge data set of customer data

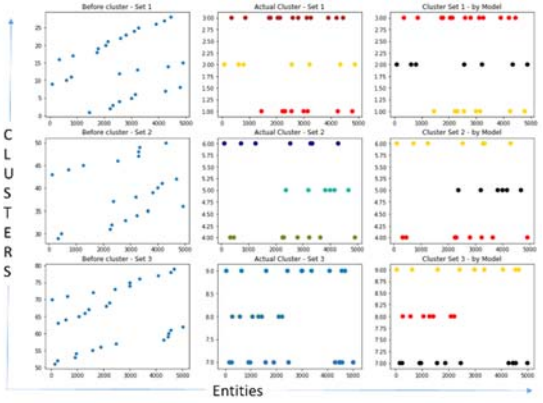

Fig. 8. Clustered data based on their similarity scores for 3 data sets

The cluster shown in Fig. 7 seems to be cumbersome and difficult to interpret as all the data points/ entities are represented in the data space together. To have a better interpretation and understanding some entities clustered and reported in Fig. 8. Each sub figures, in Fig. 8, represents 3 sets of data points/ entities within a two-dimensional space. Each set has depicted before the algorithm runs how the data is distributed, what was the actual data supposed to be clustered together and the third figure. depicts how the data is clustered. As per the Fig. 8, all three subsets are very neatly clustered the similar records as per the expectations.

We have evaluated the cluster's quality with the following two measures:

External measure: Compared the clustering of a set of data, which were prior or expert specified knowledge – with ground truth and found it is good in terms of clustering the data. The experiment seems to be found over 98% of data seems to be identified in the duplication process.

Internal measure: Evaluated the cluster quality by using cluster dendrogram and silhouette plots. As shown in Fig. 9, the dendrogram - visual representation of the compound correlation data component clusters are formed by joining individual components. From this, it can be inferred that tightly correlated clusters are nearer the bottom of the dendrogram. This diagram is a convenient way to depict the pairwise similarities or dissimilarities among the objects. So here, the level at which branches merge and it seems to be similar within the cluster and dissimilar to other clusters. The quantitative evaluation of the quality of dendrograms is usually identified with Sum-Square Error (SSE) also called leaf node error and standard deviation of SSE.

When analyzed SEE or Leaf node it is as small as close to 0.12, which means the abstraction of the quality is good. Similarly, the standard deviation of the SSE is also close to zero hence there is no low occurrence of under or over-abstraction.

The other part of Fig. 10, depicted the Silhouettes plot - which offers a unique advantage dependency on the actual partition of the data points and independent of algorithm used for making clusters[22]. The average Silhouette width is 0.52 within the set. Generally, a large overall average silhouette width can be taken; in this case, this algorithm has considered being having a strong clustering structure.

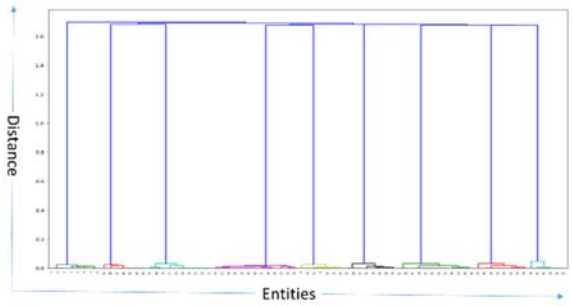

Fig. 9. Dendrogram

As shown in the Fig. 10 - Silhouette plot the average silhouette coefficients is: 0.9872986700438008, and the one set of coefficients are given below for reference:

0.98862214, 0.97530403, 0.98816568, 0.98862214, 0.98699489, 0.98862214, 0.97530403, 0.98862214, 0.98873533, 0.98862214, 0.98699489, 0.98862214, 0.97782835, 0.99433632, 0.99433632, 0.99433632, 0.99433632, 0.99433632, 0.98934239, 0.97934875, 0.98957609, 0.98957609, 0.98957609, 0.97934875, 0.98957609, 0.97620826, 0.98957609, 0.98721515, 0.98957609

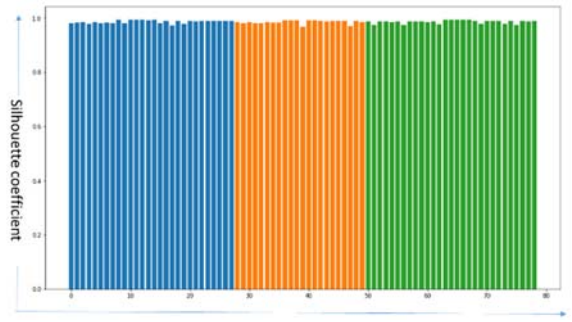

Fig. 10. Silhouette plot

As per the above silhouette coefficients, it can be observed that the quality of the cluster is very good as the average coefficient is 0.98 it is a very accurate clustering model.

6. Discussion & review of the results

Data Set: The Experimental evaluation discussed in the above sections is a structured customer data with 100000 of customer entities with customer name and address attributes. The article, shown the first 5000 records for better depiction in the diagrams otherwise all the records would be cumbersome and not legible to demonstrate.

Performance measure: The performance and effectiveness of a cluster are evaluated by two measures, such as Purity and F-measure[23].Precision is a measure of model relevancy. A recall is the measure of models completeness

F-Measure[24] of a cluster for given class i, derived as $F(i,j) = \frac{2\,p(i,j)*r(i,j)}{p(i,j)+r(i,j)}$ (3)

Where p(i,j) = $\frac{n_{ij}}{n_j}$ (4)

r(i,j) = $\frac{n_{ij}}{n_i}$

Here i is a class of cluster j and n_i is a number of CDs in class i, and n_j is a number of CDs in cluster j, n_{ij} is a number of CDs in cluster j and class i.

From this entire cluster's f-measure is calculated as below:

$$F = \sum_i \frac{n_i}{n}\ max_j(F(i,j)) \quad (5)$$

F-measure is a weighted harmonic mean – a conservative average, which is a combined measure that assesses the P/R trade-off.

The purity of a cluster is defined as follows

$$purity(j) = \frac{1}{n_j} max_j(n_j) \quad (6)$$

Similar total Custer purity[24] is

$$\text{Purity} = \sum_j \frac{n_j}{n} * purity(j) \quad (7)$$

As shown in table 9, for all three sets of data evaluated F1-scores are 1. All precision, recall values for all the tree sets are equals to one. Hence, the f-measure is very high. Hence, it is a very accurate model.

F-measure = $F = \frac{2\,(1*1)}{(1+2)} = 1$

Clusters F-measure is: 1

Purity is also calculated as 1 and hence the quality of the cluster is very high.

	Precision	Recall	F1-Score	Support
1	1.00	1.00	1.00	8
2	1.00	1.00	1.00	7
3	1.00	1.00	1.00	13
Micro avg	1.00	1.00	1.00	28
Macro avg	1.00	1.00	1.00	28
Weighted avg	1.00	1.00	1.00	28
4	1.00	1.00	1.00	8
5	1.00	1.00	1.00	6
6	1.00	1.00	1.00	8
Micro avg	1.00	1.00	1.00	22
Macro avg	1.00	1.00	1.00	22
Weighted avg	1.00	1.00	1.00	22
7	1.00	1.00	1.00	12
8	1.00	1.00	1.00	7
9	1.00	1.00	1.00	10
Micro avg	1.00	1.00	1.00	29
Macro avg	1.00	1.00	1.00	29
Weighted avg	1.00	1.00	1.00	29

Table 9: Precision, recall, f1-score and support

Both together high precision and high recall high accurate model. Hence it is concluded that this model is highly accurate. After analyzing these data, it has been observed that over 90% of duplicate records are clustered together and can be easily eliminated or linked into one hierarchal entity. Rest of the duplicates, which are small in number, were not identified by the algorithm as specific scenarios such as Post box number are given in the address as NW, SW, etc., these can be addressed by tuning the algorithm to the specific business cases

7. Conclusion

HP Algorithm is based on the cosine similarity measure, which is widely used for comparing text documents. The proposed algorithm used cosine similarities as it is easy to understand, easy to use, and also works based on the orientation rather magnitude. This article exemplifies the various reasons for data getting duplicated (customer data) and is different solutions existing in the current literature. There is no certain solution or a tool to eliminate the duplication of master data. The proposed solution with a heuristic pruning algorithm to identify and help to eliminate duplicate customer data. This solution has two steps, the first step is a time process to eliminate all duplicates in the existing data and the second step is preventing duplicate data from being created from online processing. This algorithm compares the record's relevancy with each other record and gives the score of the similarity. All similar records having the minimum threshold values are grouped into a cluster. All records in one cluster belong to one customer hierarchy are to be considered as a single entity. The matching algorithm will be input for identifying the similarities and the cosine similarity algorithm takes key inputs from the matching algorithm and finds the similarity. The time complexity of the algorithm is less compared to the regular algorithm's running time, the overall running time of the HP algorithm is reduced by $2nm\text{-}m^2$ (where n>>m.) running time. The quality of the cluster is evaluated using dendrogram's quality evaluation techniques – SSE and Std (SSE). Silhouette coefficients are ~0.98, which is very accurate. Both cluster quality measures of F-measure and purity are very high (1), hence the quality of the cluster is very high and accurate. All duplicate scenarios discussed in section 3, were addressed with this algorithm. It seems above 90% of duplicate data and duplicate scenarios have been captured and addressed by this algorithm. Since it is built on a distributed computing framework and lambda architecture, it is a scalable and reliable solution. Since industries are spending heavily on the deduplication process and there are many interesting factors in this approach, our research group working on this to further develop and tune this algorithm on a distributed framework.

8. Conflicts of Interest

The authors declare no conflict of interest.

References

[1] M. Spruit and K. Pietzka, "MD3M: The master data management maturity model," *Computers in Human Behavior,* vol. 51, pp. 1068-1076, 2015/10/01/ 2015.
[2] S. T. Ng, F. J. Xu, Y. Yang, and M. Lu, "A Master Data Management Solution to Unlock the Value of Big Infrastructure Data for Smart, Sustainable and Resilient City Planning," *Procedia Engineering,* vol. 196, pp. 939-947, 2017/01/01/ 2017.
[3] R. Bozeman, "Data Duplication and HubSpot:," in *Dealing With Duplicates and the Impact They Have on Your Business*ubSpot's, Ed., ed, Jul 11, 2018
[4] G. S. Manku, A. Jain, and A. Das Sarma, "Detecting near-duplicates for web crawling," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 141-150: ACM.
[5] A. Schofield, L. Thompson, and D. Mimno, "Quantifying the Effects of Text Duplication on Semantic Models," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2737-2747.
[6] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, 2002, pp. 380-388: ACM.
[7] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon, "Metric and kernel learning using a linear transformation," *Journal of Machine Learning Research,* vol. 13, no. Mar, pp. 519-547, 2012.
[8] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," 2010.
[9] Q. Jiang and M. Sun, "Semi-supervised SimHash for efficient document similarity search," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 93-101: Association for Computational Linguistics.
[10] E. N. Borges, M. G. de Carvalho, R. Galante, M. A. Gonçalves, and A. H. F. Laender, "An unsupervised heuristic-based approach for bibliographic metadata deduplication," *Information Processing & Management,* vol. 47, no. 5, pp. 706-718, 2011/09/01/ 2011.
[11] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and efficient fuzzy match for online data cleaning," presented at the Proceedings of the 2003 ACM SIGMOD international conference on Management of data, San Diego, California, 2003.
[12] W. W. Cohen, "Data integration using similarity joins and a word-based information representation language," *ACM Trans. Inf. Syst.,* vol. 18, no. 3, pp. 288-321, 2000.
[13] C. F. Dorneles, C. A. Heuser, A. E. N. Lima, A. S. d. Silva, and E. S. d. Moura, "Measuring similarity between collection of values," presented at the Proceedings of the 6th annual ACM international workshop on Web information and data management, Washington DC, USA, 2004.
[14] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of Jaccard coefficient for keywords similarity," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2013, vol. 1.
[15] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *KDD workshop on text mining*, 2000, vol. 400, pp. 525-526: Boston.
[16] V. Thada and V. Jaglan, "Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for web retrieved documents using genetic algorithm," *International Journal of Innovations in Engineering and Technology,* vol. 2, no. 4, pp. 202-205, 2013.
[17] A. Huang, "Similarity measures for text document clustering," in *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, 2008, pp. 49-56.
[18] D. Melo *et al.*, "Hierarchical Density-Based Clustering Based on GPU Accelerated Data Indexing Strategy," *Procedia Computer Science,* vol. 80, pp. 951-961, 2016/01/01/ 2016.
[19] F. S. Al-Anzi and D. AbuZeina, "Toward an enhanced Arabic text classification using cosine similarity and Latent Semantic Indexing," *Journal of King Saud University - Computer and Information Sciences,* vol. 29, no. 2, pp. 189-195, 2017/04/01/ 2017.
[20] P. Xia, L. Zhang, and F. Li, "Learning similarity with cosine similarity ensemble," *Information Sciences,* vol. 307, pp. 39-52, 2015/06/20/ 2015.

[21] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures," presented at the Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, D.C., 2003.
[22] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics,* vol. 20, pp. 53-65, 1987/11/01/ 1987.
[23] S. Eddamiri, E. M. Zemmouri, and A. Benghabrit, "An improved RDF data Clustering Algorithm," *Procedia Computer Science,* vol. 148, pp. 208-217, 2019/01/01/ 2019.
[24] Z. Abedjan, "Improving RDF data with data mining," 2014.

Authors Profile

Kari Venkatram, Completed Masters in Software Engineering from BITS, Pilani. Perusing PhD School of Computer Science and Engineering, Vellore Institute of Technology, Vellore – 632 014, Tamil Nadu, India. His research interests are in Master Data Management and Big Data Analytics. He is working on some of the innovative approach to solve practical challenges in MDM with help of Artificial Intelligence and Machine Learning models along with Big data solutions.

Geetha Mary A received her Ph.D. in Computer Science Engineering from Vellore Institute of Technology, Vellore, India. She has completed M.Tech in Computer Science and Engineering from VIT and B.E. from University of Madras, Tamil Nadu, India. She is working for VIT as Associate Professor. She was awarded Merit Scholarship for her best academic performance for the year 2007-2008 during her M.Tech study. She has authored more than 50 journal and conference papers. She has authored book chapters in the area of data mining and artificial intelligence. Her research interests include security for data mining, databases and intelligent systems. She works to empower healthcare management using computer science. Dr. Geetha Mary is associated with many professional bodies like IACSIT, CSTA and IAENG.