CHAPTER 2

REVIEW OF LITERATURE

Data cleaning is the process of detecting and eliminating duplicate records. The data cleaning of large databases of information needs to be processed as quickly, efficiently and accurately as possible. Detection and elimination of duplicate data is the major research area in the data warehouse. Several existing methods are available for data cleaning. But the existing methods are costly and will take a very long time for cleaning large databases. At the same time, some of the existing methods are well suited for only particular types of errors. The goal of this research work is to improve the quality of the data and increase speed of the data cleaning process. In this research work, a framework is developed to reduce the number of false positives, to speed up the data cleaning process, reduce the complexity and to improve the quality of data.

To justify the problem statement given in Chapter 1 (page no. 12), the review of literature is carried out on the following parameters:

i. Attribute Selection

ii. Token Formation

iii. Blocking records

iv. Similarity functions for Record Linkage

v. Duplicate Detection and Elimination
The existing works and performance of each method which is involved in this research work is explained and summarized below.

2.1 Attribute Selection

Attribute selection is the process of identifying best and suitable attributes for the data cleaning process. It is a preprocessing step in detection and elimination of duplicate data by reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving result comprehensibility. However, the recent increase of data poses many problems in data quality with respect to efficiency and effectiveness. Identification of duplicate data with large amount of data is difficult and it takes more time to complete the data cleaning process. As well as, process of detection and elimination of duplicate data mainly depends on the attribute selection and identification of duplicates which are varied based on the selection of attributes. For example, detection of duplicates is poor with age, salary and high with SSN number, name and address. Hence, there is a need to select best and suitable attributes for duplicate data detection and elimination. The main advantage of attribute selection is that it identifies suitable attributes for data cleaning and increases the speed and improves the quality of the data.

In this research work, attribute selection algorithm is used to select suitable attributes from data warehouse for the duplicate detection and elimination process. This attribute selection algorithm is used for identifying best attributes to improve the quality of data by detecting exact and inexact duplicates. Related attribute selection algorithms
in data mining which are used for identifying best attributes to get better analysis result are discussed below.

Kira et al presents an efficient feature selection algorithm, called RelieF, which is used for evaluating each attribute by its ability to distinguish among instances that are near each other [KR, 92]. Their selection criterion, the feature relevance, is applicable to numeric and nominal attributes. The greatest limitation of RelieF is its inability to identify redundant features within a set of relevant features. Consequently, the set of features selected by RelieF may not be optimal.

John et al proposes the greedy algorithm, that is enhanced based on the wrapper model and introduces an information-theoretic method for selecting relevant features [JKP, 94]. In their Mutual-Information-Greedy (MIG) Algorithm defined for Boolean noise-free features, the feature is selected if it leads to the minimum conditional entropy of the classification attribute. Since the data is assumed being noise-free, no significance testing is required (any non-zero entropy is significant). The above assumptions leave the MIG algorithm at quite a distance from most practical problems of reducing data warehouse dimensionality. The feature filter model assumes filtering the features before applying a data mining algorithm, while the wrapper model uses the data mining algorithm itself to evaluate the features. The possible search strategies in the space of feature subsets include backward elimination and forward selection. The performance criterion of the wrapper model is the prediction accuracy of a data mining algorithm, estimated by n-fold cross validation.
Liu and Motoda developed a unified model of the feature selection process that includes four parts: feature generation, feature evaluation, stopping criteria, and testing [LM, 98]. In addition to the “classic” evaluation measures (accuracy, information, distance, and dependence) that can be used for removing irrelevant features, they mention important consistency measures (e.g., inconsistency rate), required to find a minimum set of relevant features. By decreasing the inconsistency rate of data, both irrelevant and redundant features are removed. However, consistency measures are only suitable for selecting discrete features.

Luigi Portinale and Lorenza Saitta proposed AdHoc method for feature selection that comprises two main steps, namely the Data Reduction step and the Feature Selection step [LL, 02]. The Data Reduction step is concerned with reducing the dimensionality of the data and is independent of the knowledge discovery algorithm. Data reduction is beneficial in two points of views: on the one hand, it provides a deep insight into the structure of the problem at hand (each factor represents a data dimension); on the other hand, it delivers a very good starting point for the search of relevant features. Indeed, in the second step of AdHoc, namely the Feature Selection step, a genetic algorithm (GA) is used to select the minimum number of most informative features from every factor.

Magnus Erick and Hvass Pedersen say, an Embedded feature selection, wrapper approach and filtering methods are commonly used in feature selection depending on the complexity of the data mining task [MH, 03]. An Embedded feature selection is a part of
the data mining process, in that features are added or removed while building the model, depending on the change in accuracy. The wrapper approach uses data mining as a sub-routine, evaluating the quality of the feature subset by the accuracy of the model discovered in the data mining sub-process. Filtering is a non-recurring preprocessing step used when the data mining task is too expensive for iterative use as in the wrapper method, and a low-cost approximation is unavailable or unviable.

Daniel et al says the feature selection method can be divided into two stages [DLD⁺, 06]. The first stage consists of generating a number of feature sets of fixed size, then running a 10 fold cross validation using only the features found in these sets, to determine how well they are able to classify the data. The sets of features are then sorted by a given criteria and the best of these randomly generated sets are selected for the second stage of the algorithm. For the second stage of the method have a number of ranked feature sets. Using these sets, a new set composed of the union of the features found in the selected sets is created.

The above addressed feature selection algorithms are used to select the attributes in data mining to improve the efficiency of the analysis result. The problems of existing approaches on attribute selection are reviewed. In these existing feature selection algorithms, criteria which are applicable for data cleaning are not included. There are three criteria which are very important in attribute selection for data cleaning – identifying key attributes, classifying attributes with high distinct value and low missing value and measurement types of the attributes. The uniqueness of data is very important
for duplicate data detection. The proposed algorithm uses specified criteria which are more important in duplicate data detection. Because it is mainly developed to select relevant attributes for the data cleaning process which reduces the time and increase the speed.

### 2.2 Token Formation

Mainly, data is duplicated because of the typing errors and misuse of abbreviations. In detection of duplicate data, the records are scanned by comparing whole records to determine matching records in integrated data. Determining optimal record match score threshold is very hard and takes long time with large amount of data. Also, direct long record string comparison is highly inefficient and intolerant to typing errors.

In this research, token based similarity computation is used to reduce the need of long string comparison for increasing the speed of the data cleaning process. Token formation is the process of defining tokens for records using the selected attribute field values. Token formation algorithm is used to remove unimportant characters and to define token for each field values based on the types of data. The token is formed for each selected attribute field which is having the highest rank. Related works and importance of tokens are addressed below.

Hernandez and Stolfo said that the large data file is sorted on designated fields to bring potentially identical records together [HS, 95], [HS, 98]. However, sorting is based
on “dirty” fields, which may fail to bring matching records together, and its time complexity is quadratic in the number of records. This sorting technique is inefficient while dealing with large data file. The merge/purge problem in a large database is solved by forming keys from some selected fields, sorting the entire data set on the keys, clustering the sorted records and using a scanning window of a fixed size to reduce the number of comparisons. Record comparison is still based on the original dirty records.

Several steps are used to clean the data warehouse [LHT+, 99]. The first step is Scrub dirty data fields. This step attempts to remove typographical errors and abbreviations in data. The second step is Sort tokens in data fields. Characters in a string can be grouped into meaningful pieces, called tokens, which are then sorted. The third step is Sort records. This step sorts the records based on the token value. The fourth step is Comparison of records. A window of fixed size is moved through the sorted records to limit the comparisons for matching records. Field weightage is used to compute the degree of similarity between two records. The final step is merging the matching records. Matching records are treated as a partial source of information and merged to obtain a record with more complete information.

In the existing approach, the token is formed and the records are sorted based on the similarity of any one of the attributes. Formed tokens are used only for sorting records in the reviewed papers.

In this research work, the simple token-formation algorithm is proposed with existing properties in forming tokens for sorting and similarity computation. The
method presented in [HS, 95], [HS, 98] would not be efficient because of wrong selection of attributes. The token based approach is used in selected attribute fields only. Hence, there is a need to select attributes before the token formation and this selected attributes should be efficient in handling duplicate data. This inefficiency is handled by selecting appropriate attribute.

2.3 Blocking Records

Blocking methods are used to group the records based on the similarity of high power attribute field values. Blocking methods select a set of candidate duplicates out of the set of possible pairs. This is necessary to reduce the number of pairs to be examined in the expensive matching phase. In general, each record has to be compared with all the remaining records to calculate similarity values of each record in record comparison. So, the record comparison takes a long time to calculate similarity computation. The main advantage of blocking method is that it reduces the time taken for the record comparison by dividing the records based on the similarity value. The main issue for a blocking algorithm is therefore a small set of resulting candidate duplicates, but additionally a very high duplicate detection ratio and efficiency is important.

In this research work, dynamic window size blocking method is used for blocking the records based on the similarities of the block-token-key. This block-token-key is generated automatically from the formed token values in order to reduce false mismatches. Several blocking techniques are listed and explained below.
2.3.1 Standard Blocking

Rohan Baxter et al say that Standard Blocking (SB) method cluster records into blocks where they share the identical blocking key [RPT, 03]. A blocking key is defined to be composed from the record attributes in each data set. For example, the blocking key is generated by taking first four or three characters of a name attribute. A blocking key can also be composed of more than one attribute. The resulting total number of record pair comparisons of the standard blocking is $O(n^2/b)$.

2.3.2 Sorted Neighborhood Method (SNM)

Mauricio and Stolfo present one of the most common duplicate detection methods, which is the Sorted Neighborhood method. The Sorted Neighborhood method sorts the records based on a sorting key (SK) and then moves a window called Sliding window (SW) of fixed size $w$ sequentially over the sorted records. Records within the window are then paired with each other and included in the candidate record pair list. The use of the window limits the number of possible record pair comparisons for each record to $2w−1$ [HS, 98], [JV, 07]. The resulting total number of record pair comparisons of the sorted neighborhood method is $O(wn)$. 
One problem with the sorted neighborhood method is, if a number of records larger than the window size have the same value in a sorting key, similar to standard blocking, it is advantageous to do several passes (iterations) with different sorting keys and a smaller window size than one pass only with a large window size [JV, 07]. The effectiveness of this approach depends on the quality of key chosen to do the sorting [Che, 02].

3.3 Multi-pass SNM

Chen Shengxin says that the multi-pass sorted neighborhood method involves using multiple passes of the initial sorted neighborhood method [Che, 02]. It is executed in several independent runs of SNM, each time using a different key and a relatively small window [LSS+, 02]. In general, no single key will be sufficient to catch all matching records [MMT+, 03]. During each pair, the three steps - create keys, sort data
and merge - are performed selecting a different key field for each pass. Once the
duplicate records are identified during a pass one of the duplicates is eliminated from the
data source. This method provides better results than the SNM method.

![Sliding Window Diagram](image)

**Figure 2b: Sliding Window**

### 2.3.4 Clustering SNM

Feekin and Chen proposed Clustering SNM method, which first partitions the
database into independent clusters using a key extracted from the data [FC, 00]. Then
SNM is applied to each individual cluster independently. There are two phases in
Clustering SNM:

i. Cluster data

Scan the records in sequence to extract a key for each record; then using the
extracted key, partition the records into independent subsets of data (clusters).
ii. SNM

Apply SNM independently on each cluster. The key does not need to be recomputed, the key extracted above can be used for sorting.

2.3.5 Duplicate Elimination SNM (DE-SNM)

Sung et al present Duplicate Elimination SNM, which sorts the records on a chosen key and then divides the sorted records into two lists. i) a duplicate list ii) a non-duplicate list [SLL, 03]. A small window scan is first performed on the duplicate list to find the list of matched and unmatched records. The list of unmatched records is merged with the original non-duplicate list and a second window scan is performed [LSS+, 02]. Using Sorted Neighborhood method the exact or very closely matching duplicate records with equivalent keys can be identified and it would likely be more efficient to find these records first during the sorting phase [MMT+, 03]. This method removes other duplicate records and retains only one representative member of this set of duplicates by the rule set against other records with different keys. DE-SNM does not contribute much on the improvement of accuracy of SNM. The benefit of DE-SNM is that it runs faster than SNM under the same window size, especially for the databases that are heavily dirty. If the number of records in duplicate list is large, DE-SNM will run faster than SNM.

2.3.6 SNM – Priority Queue

Li Zhao et al say the algorithm scans the database sequentially and determines whether each record is scanned or not and also determines whether a member of the
cluster is represented in priority queue [LSS⁺, 02]. This queue contains a fixed number of record sets, which contain similar records that can be paired as candidate duplicates. The sorted list of all records is sequentially scanned and every record Rj is compared with the members Ri in the record set with the highest priority in the queue. If the comparison yields a distance below some threshold T1, the record Rj is included into the set of Ri, if the distance is higher than some threshold T2 this record set is skipped and the comparison is continued with the next highest priority set in the queue. If no set is found at all, a new singleton set is put into the queue. In order to reduce the number of comparisons, the record sets are pruned, i.e. they only contain ”representative” members of the set, in particular, non-representative members are those, which are very similar to other members. However, the authors do not make it clear how to find the two thresholds and how to decide if a record is “representative” [AC, 96].

2.3.7 K-way Sorting Method

The Sorted Neighborhood Method is not effective where the data source does not contain a primary key field of reference. The k-way sort duplicate detection method is designed to effectively detect duplicates in data sources that do not have any uniquely distinguishing data fields [Che, 02], [FC, 00]. This K-way sorting method follows the following steps.

The steps are,

i. Let k be the number of columns to be used for sorting
ii. Select the k-most meaningful combinations of sort keys based on the k selected columns.

iii. Assign a record identifier to each record.

iv. Sort records based on the selected sort key combination

v. For each sorted set of data, compare adjacent rows within a given window size

vi. Draw k graphs for each sort

vii. Examine the k graphs collectively. If matches occur between some records or identifier exceeds certain threshold, then it should be mapped into the summation graph.

viii. The summation graph should handle transitive closure.

2.3.8 Bigram Indexing

Rohan Baxter proposes the Bigram Indexing (BI) method as implemented in the Febrl record linkage system which allows for fuzzy blocking [RPT, 03]. The basic idea is that the blocking key values are converted into a list of bigrams (sub-strings containing two characters) and sub-lists of all possible permutations will be built using a threshold (between 0.0 and 1.0). The resulting bigram lists are sorted and inserted into an inverted index, which will be used to retrieve the corresponding record numbers in a block.

The number of sub-lists created for a blocking key value both depends on the length of the value and the threshold. The result of bigram list in smaller blocks will be produced in the inverted index if the value of threshold is low and the sub-list is small
with per blocking key value. In the information retrieval field, bigram indexing has been found to be robust to small typographical errors in documents. Like standard blocking, the number of record pair comparisons with two data sets with \( n \) records each and \( b \) blocks, all containing the same number of records is \( O(n^2 b) \) [MA, 07]. The number of blocks \( b \) will be much larger in bigram indexing.

### 2.3.9 Canopy Clustering with TF-IDF

Markus Doring and Anreas Millar explain Canopy Clustering with TF-IDF (Term Frequency-Inverse Document Frequency) forms blocks of records based on those records placed in the same canopy cluster [RPT, 03], [MA, 07]. A canopy cluster is formed by choosing a record at random from candidate set of records. This canopy clustering method uses the TF-IDF distance metric instead of bigrams [RPT, 03]. The total number of record pair comparisons resulting from canopy clustering is \( O(fn^2/c) \) where \( n \) is the total number of records, \( f \) is the average number of canopy and \( c \) is the number of canopies. The threshold parameter should be set so that \( f \) is small and \( c \) is large, in order to reduce the amount of computation. However, if \( f \) is too small, then the method will not be able to detect typographical errors.

### 2.3.10 SNM-IN

The main difference between SNM and SNM-IN is as follows: while SNM compares the new record entering the current window with all previous records in the window, SNM-IN will first check whether the new record with previous records are
duplicate or non-duplicate with the new record, there is no need to compare them with the new record [LSS⁺, 02]. Thus, with Lower bound and Upper bound, SNM-IN will reduce a lot of comparisons.

### 2.3.11 SNM-INOUT

SNM-IN follows SNM with only one anchor record and SNM-INOUT follows SNM with two anchor records [LSS⁺, 02]. One anchor record is an inAnchor record and the other is outAnchor record. SNM-INOUT has some changes in Merge phase of SNM method. When a new record enters the current window, it is first compared with the two anchor records, which introduces one more comparison than SNM-IN. However, in SNM-INOUT, for each record in the current window, the chance of determining the new record as duplicate record or not is increased, which produces more chance to reduce comparisons than SNM-IN does. Since the outAnchor is outside the window and the last record in the window will compare with it, SNM-INOUT will obtain a few more duplicate pairs than SNM-IN if both run at the same window size.

### 2.3.12 Disjunctive Blocking

Bilenko et al present Disjunctive blocking selects record pairs that are placed in the same block by at least one blocking predicate from the selected subset of predicates [BKM, 06]. This strategy can be viewed as selecting pairs for which a disjunction of predicates evaluates to true. The blocking function is trained by selecting a subset of blocking predicates.
2.3.13 Disjunctive Normal Form (DNF) Blocking

Bilenko et al present DNF blocking which selects object pairs that satisfy at least one conjunction of blocking predicates from a selected subset of conjunctions [BKM, 06]. This strategy can be viewed as selecting pairs for which a disjunction of predicate conjunctions evaluates to true. The blocking function is trained by constructing the DNF formula over the blocking predicates.

2.3.14 Fuzzy Blocking

Jordi Nin and Vicen said that in the standard blocking method, the blocking keys are selected from the noisy attributes [JV, 07]. The fuzzy blocking method substitutes the blocking key by the results of an OWA (Ordered Weighting Averaging) operator. This fuzzy blocking method follows the following steps:

i. A data in the data set is normalized

ii. A Fuzzy quantifier is selected

iii. OWA operator with the fuzzy quantifier

iv. The OWA result is rounded and the record is blocked using this value as a blocking key.

The fuzzy blocking method has two parameters: The OWA quantifier that decides the way to aggregate, and the rounding method that decides the number of blocks to do.
2.3.15 Adaptive Sorted Neighborhood Method (A-SNM)

Su Yan et al said that Adaptive Sorted Neighborhood Method uses the original SNM approach to slide a fix-sized window $n - w + 1$ times, to generate $n-w+1$ number of blocks when applied to $n$ records [SDM*, 07]. Note that these generated blocks are overlapping each other, and fix-sized. However, in the adaptive SNM, the “window” becomes only an interim tool to generate final blocks, and all generated blocks will have different (thus adaptive) sizes.

The above discussed blocking methods available in the literatures are used to block the records to reduce the number of comparisons for similarity computation. In the existing blocking method, records are blocked based on the some attribute field values by selecting first three characters from each field. Blocking with poor quality data is worst and produces inaccurate result. In this research work, blocking key is generated using token values and token based blocking method is introduced to reduce the time taken for the comparison process and to improve the accuracy of the data by reducing false mismatches.

2.4 Similarity functions for Record Linkage

Record linkage algorithms are depending on string similarity functions to find discriminating values between equivalent and non-equivalent record fields [Mik, 03]. There are two types of similarity function to calculate similarity value between strings: sequence-based functions and token-based functions. Sequence-based functions model is
a fundamental model to compare two strings and this model is mainly used in the long string value. Token-based functions are used to compare the token values which are formed for each record field.

Similarity computation is very important in duplicate data detection. In record linkage, similarity functions are used for finding threshold value for each record to check whether record is duplicated or not. The basic goal of similarity computation is to classify each record pairs as matching or non-matching. Data warehouse frequently contain field values and records that refer to the same entity but are not syntactically identical. This problem is typically handled during a tedious manual data cleaning, or “de-duping”, process. Because similarity measures between strings may fail to identify exact or inexact (slightly changed duplicate value) duplicate because of the systematic typographical errors. In this research work, token based similarity computation is followed to avoid the problem of long string similarity computation. At the token level, certain words are not needed when comparing two strings for equivalence. Before the token formation, unwanted or unimportant characters are ignored and certain characters are consistently replaced by other characters such as expansion the abbreviations. The existing methods are surveyed and listed below which are used in string and token based similarity computation.
2.4.1 Sequence-based String Similarity Functions

Hamming Distance

Mohamed et al. discussed that this is defined as the number of bits which differ between two binary strings i.e. the number of bits which need to be changed to turn one string into the other [MVA, 02]. The simple hamming distance function can be extended into a vector space approach where the terms within a string are compared, counting the number of terms in the same positions. The Hamming distance is used primarily for numerical fixed size fields like Zip Code or SSN. It counts the number of mismatches between two numbers.

String Edit Distance

Mikhail Bilenko says the Hamming distance cannot be used for variable length fields since it does not take into account the possibility of a missing letter [MVA, 02], [Mik, 03], [UMR, 02]. The most well-known sequence-based string similarity measure is string edit distance, also known in its simplest form as Levenshtein distance. The edit distance between two strings is the minimum cost to convert one of them to the other by a sequence of character insertions, deletions, and replacements. Each one of these modifications is assigned a cost value function.

String Edit Distance with Affine Gaps

The extended model of the edit distance function is string edit distance with Affine Gaps [Mik, 03]. This model is introduced to allow contiguous sequences of
mismatched characters, or gaps, in the alignment of two strings. Most commonly the gap penalty is calculated using the affine model, which assigns a high penalty for starting a gap, and a lower linearly increasing penalty for extending a gap. Different edit operations have varying significance in different domains.

**Soundex**

Mohamed et al tells the purpose of the Soundex code is to cluster together names that have similar sounds [MVA, 02]. This soundex code function assigns the numeric value for each character. The Soundex code of a name consists of one letter followed by three numbers.

**Jaro**

Jaro proposed a metric that is less sensitive to character transpositions [MVA, 02]. Jaro introduced a string comparison function that accounts for insertions, deletions, and transpositions. Jaro’s algorithm finds the number of common characters and the number of transposed characters in the two strings. A common character is a character that appears in both strings within a distance of half the length of the shorter string. A transposed character is a common character that appears in different positions.
Jaro-Winkler String Similarity

The Jaro-Winkler distance is a measure of similarity between two strings. It is designed and best suited for short strings. It is a variant of the Jaro distance metric and mainly used in the area of record linkage (duplicate detection) [Mik, 03].

The basis of this comparator is the count of common characters between the strings, where a character is counted as common if it occurs in the other string within a position distance that depends on the string length. The value of the string comparator is further determined by a count of transpositions. Transpositions are determined by pairs of common characters out of order. A further enhancement to the score computation due to Winkler is based on the observation that typographical errors occur more commonly toward the end of a string, so that strings that agree on prefixes (1 to 4 characters) are given a higher comparison score. Specifically, the algorithm considers the first four prefixes of each string and lets $p$ be the length of the longest prefix pair that agree exactly ($a_i = b_i$).

2.4.2 Token-based String Similarity Functions

Jaccard Coefficient

Mikhail Bilenko says that Jaccard similarity can then be used as the simplest method for computing likeness as the proportion of tokens shared by both Strings [Mik, 03]. The primary problem with Jaccard similarity is that it does not take into account the relative importance of different tokens.
TF-IDF cosine similarity

Miktail Bilenko says that the Term Frequency-Inverse Document Frequency (TF-IDF) cosine similarity is a vector-based measure of the similarity of two strings [Mik, 03]. The basic idea behind cosine similarity is to transform each string into a vector in some high dimensional space such that similar strings are close to each other. There are many ways of transforming a string in the database into a vector. The TF-IDF vector is a popular choice for this representation. The TF-IDF vector is composed of the product of a term frequency and the inverse document frequency for each token that appears in the string. TF-IDF cosine similarity is computationally efficient due to high sparsity of most vectors, and provides a reasonable off-the-shelf metric for long strings and text documents.

The TF-IDF weighting scheme is useful for similarity computations because it attempts to give tokens weights that are proportional to the tokens’ relative importance. However, the true contribution of each token to similarity is domain-specific.

n-grams

Tokenization is typically performed by treating each individual word of certain minimum length as a separate token, usually excluding a fixed set of functional “stopwords”. An alternative tokenization scheme is known as n-grams: it relies on using all overlapping contiguous subsequences of length n as tokens.
N-grams is another approach for computing the distance between two strings. The N-grams comparison function forms the set of all the substrings of length \( n \) for each string. The distance between the two strings is defined as \( \Sigma -x \ a \ b f \left( x \right) f \left( x \right) \) where \( f \left( x \right) a \) and \( f \left( x \right) b \) are the number of occurrences of the substring \( x \) in the two strings \( a \) and \( b \), respectively. Bigrams comparison \( ( n = 2 ) \) is known to be very effective with minor typographical errors. It is widely used in the field of information retrieval [Mik, 03], [MVA, 02]. Trigrams comparison \( ( n = 3 ) \) is used by Hylton in record linkage of bibliographical data. Most recently, N-grams was extended to what is referred to as Q-grams for computing approximate string joins efficiently. N-grams is more efficient than edit distance or Jaro’s algorithm in the case of strings that contain multiple words and are known to be commonly in error with respect to word order.

**Vector-space Similarity Functions**

In Euclidean space the Minkowski family of metrics, also known as the \( Lk \) norms, includes most commonly used similarity measures for objects described by \( d \)-dimensional vectors [Mik, 03], [BM, 02]. The \( L2 \) norm, commonly known as Euclidean distance, is frequently used for low-dimensional vector data. Its popularity is due to a number of factors:

- Intuitive simplicity: the \( L2 \) norm corresponds to straight-line distance between points in Euclidean space;
- Invariance to rotation or translation in feature space;
• Mathematical metric properties: non-negativity \((L_2(x, y) \geq 0)\), reflexivity \((L_2(x, y) = 0 \iff x = y)\), symmetry \((L_2(x, y) = L_2(y, x))\), and triangle inequality \((L_2(x, y) + L_2(y, z) \geq L_2(x, z))\).

Despite these attractive characteristics, Euclidean distance as well as other Minkowski metrics suffers from the *curse of dimensionality* when they are applied to high-dimensional data. As the dimensionality of the Euclidean space increases, sparsity of observations increases exponentially with the number of dimensions, which leads to observations becoming equidistant in terms of Euclidean distance.

<table>
<thead>
<tr>
<th>S. No</th>
<th>Similarity Function</th>
<th>Type of Data</th>
<th>Year</th>
<th>Author Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hamming Distance</td>
<td>Numerical</td>
<td>1950</td>
<td>Richard Hamming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fixed size</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Field, equal length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>String Edit Distance</td>
<td>Long String, arbitrary length</td>
<td>1965</td>
<td>Levenshtein</td>
</tr>
<tr>
<td>3</td>
<td>String Edit Distance with Affine Gaps</td>
<td>Long String</td>
<td>1970</td>
<td>Needleman &amp; Wunsch</td>
</tr>
<tr>
<td>4</td>
<td>Soundex</td>
<td>String</td>
<td>1918</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Jaro</td>
<td>Short String</td>
<td>1989</td>
<td>Jaro</td>
</tr>
<tr>
<td>6</td>
<td>Jaro-Winkler String Similarity</td>
<td>Short String</td>
<td>1999</td>
<td>Winkler</td>
</tr>
<tr>
<td>7</td>
<td>Jaccard</td>
<td>Token String</td>
<td>1901</td>
<td>Jaccard</td>
</tr>
<tr>
<td>Coefficient</td>
<td>Similarity Functions</td>
<td>Token String</td>
<td>1983</td>
<td>Fellegi and Sunter</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
<td>--------------</td>
<td>------</td>
<td>-------------------</td>
</tr>
<tr>
<td>8</td>
<td>Vector-space</td>
<td>Token String</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Similarity Functions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TF-IDF cosine</td>
<td>Token String</td>
<td>1983</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>similarity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>n-grams</td>
<td>Token String</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>Fellegi and Sunter</td>
<td>Token String</td>
<td>1969</td>
<td>Fellegi and Sunter</td>
</tr>
</tbody>
</table>

*Table 2a: Comparison of Similarity Functions*

**Fellegi and Sunter**

Fellegi and Sunter gave a formal model for record linkage that involves optimal decision rules that divide a product space $A \times B$ of pairs of records from two files $A$ and $B$ into matches and non-matches, denoted by $M$ and $U$, respectively [BM, 02], [FS, 69], [Wil, 93]. The main issue is the accuracy of estimates of probability distributions used in a crucial likelihood ratio. When estimates are sufficiently accurate, decision rules are (nearly) optimal. The optimality is in the sense that, for fixed bounds of the proportions of false matches, the size of the set of pairs on which no decision is made is minimized.

The comparison of similarity function is described in Table 2a. This table shows that what kind of data is supported by each similarity functions. This table also contains list of string similarity functions and token-based similarity functions.
Several string based similarity computation methods are available for string comparison to identify whether record pairs are duplicate or not. In this research work, Jaccard similarity method is used for token-based similarity computation. Because Jaccard similarity algorithm is a very simple matching algorithm for token-based approach.

2.5 Duplicate Detection and Elimination

A good duplicate detection algorithm should identify all the duplicates which exist in the dataset. Duplicate detection is the problem of detecting multiple records, which describes the same real world entity. Duplicate detection method should be efficient to identify records that are not exactly duplicates. Duplicate elimination is hard because it is caused by several types of errors like typographical errors, and equivalence errors—different (non-unique and nonstandard) representations of the same logical value. Also, it is important to detect and clean equivalence errors because an equivalence error may result in several duplicate records. In duplicate elimination process, only one copy of duplicate data should be retained by ignoring other duplicate values.

In this research work, rule based duplicate detection and elimination approach is used to detect and eliminate duplicate values. Duplicate data identification rule is developed to identify duplicate values using certainty factor and threshold value. Duplicate data elimination rule is developed to eliminate duplicate data by identifying
importance of attributes to retain only one copy of exact duplicate data. The existing methods of duplicate data detection and elimination are described below.

2.5.1 Transitive Closure

Transitive closure approach is used to reduce the time for duplicate detection. If records $a$ and $b$ are found to be similar and, at the same time, records $b$ and $c$ are also found to be similar, the transitive closure step can mark $a$ and $c$ to be similar if this relation was not detected by the equational theory [Par, 06]. Moreover, records $a$ and $b$ must be within $w$ records to be marked as similar by the equational theory. The same is true for records $b$ and $c$. But, if the transitive closure step is used, $a$ and $c$ need not be within $w$ records to be detected as similar.

2.5.2 Graph-based Decision Models

Graph-based decision models also build on the idea that taking relationships between different objects into account increases the overall duplicate detection accuracy [Par, 06]. In contrast to the relationship-based similarity measures, these decision models build up graph structures between the objects and then either try to propagate information about inferred duplicates through the graph or find some global optimal solution.

The graph-based decision models are very interesting and promising approaches, but they still suffer from the complexity problem of building up such a graph-based data
structure and the computational problem of calculating a globally optimal solution. The use of a transitive closure approach is not suitable for detecting duplicate data in single pass SNM. In this research work, a rule based approach is used to eliminate the duplicates by retaining only one copy of duplicate by using threshold value and certainty factor.

Mong Li Lee et al paper on, “Cleansing Data for Mining and Warehousing”, examined the problems of detecting and removing duplicate records [LHT+, 99]. The process and drawbacks of the existing methods for data cleaning is discussed. The existing methods are SNM and DE-SNM. He found that there is a need to bring potentially matching records to a close neighbourhood for duplicate record detection. So the main objective of this paper is to bring potentially matching records to a close neighbourhood. This will enable more matching records to be detected and removed. In this research paper, a method is proposed to determine the degree of similarity between two records by using field weightage. This approach to cleaning a database comprises of five steps. i. Scrubbing data fields using external source file to remove typographical errors and use the abbreviations. ii. Sorting the tokens in the data fields iii. Sorting the records in the database. iv. Comparing the records by using fixed size window to limit the comparisons for matching records. v. Merging and marching records.

There are two points to be considered from the above paper for this research work. They are criteria for forming tokens and identifying the highest criteria among the fields and reducing the false-positive percentage. In this research work, a token based
methodology is adopted for detection and elimination of duplicate records. Attribute selection algorithm is proposed and implemented to identify best and suitable highest criteria attributes for duplicate data detection.

Sam Y.Sung Zhao and Li, Peng Sun paper on, “A Fast Filtering Schema for Large Database Cleansing”, presents a simple and fast comparison method, TI Similarity, which reduces the time for each comparison [SLS, 02]. A new detection method RAR is proposed to further reduce the number of comparisons. With RAR and TI similarity, this approach for cleansing large databases is composed of two processes: Filtering process and Pruning process. In filtering process, a fast scan on the database is carried out with RAR and TI-Similarity. In pruning process, the duplicate result from the filtering process is pruned to eliminate the false positives using more trustworthy comparison methods. Related works such as SNM, Clustering SNM, Multi-pass SNM, Equational theory, Record Similarity (RS) and Edit Distance are discussed along with comparisons between methods. Existing data cleansing methods are costly and will take very long time for cleansing large databases. Large proportion of time in data cleansing is spent on the comparisons of records. TI similarity is a simple and fast comparison method. It computes the field’s weights and filed similarity and the degree of similarity for records. RAR (Reduction using Anchor Record) is a fast detection method. It is also “Sorting and the merging” based and thus can be summarized in the three phases: Create key, Sort data and Merge. The previous two phases are the same as those in SNM. RAR uses the D-rule and ND-rule to reduce unnecessary comparisons. Finally,
comparison of the performance between proposed methods with existing methods is
given along with diagrammatical representation.

From this research paper, the field matching and records matching are identified
as very much important in similarity computation. There are two points addressed and
considered in this research work from the above paper. They are logic operations and
certain constraints have to be applied in field and record matching and elimination of
noisy record. In this research paper, token based jaccard similarity function is applied in
field and record similarity to calculate similarity matching threshold values for each pair
of records and stored in LOG table.

Christie I. Ezeife and Timothy E. Ohanekwu, in their “Use of Smart Tokens in
Cleaning Integrated Warehouse Data”, talk about identifying common errors which
occurs during record integration in the data warehouse [EO, 05]. Two algorithms are
proposed, the first algorithm uses smart tokens defined from integrated records to match
and identify duplicate records during initial warehouse cleaning. The second algorithm
uses defined tokens in incremental cleaning during warehouse refreshing. A new method
is proposed for defining smart tokens composed from most important fields of records,
which are effectively used for identifying duplicate records in data warehouses and other
records. Different kinds of rules are applied for forming tokens from alphabetic, numeric
or alphanumeric to define a smart token. Related works such as SNM, equational theory,
and field matching algorithms are discussed. Examples for data warehouse schema with
dirty records and fields are explained along with the cleaning task which can be applied
on the dirty table. The steps and the corresponding examples of each step of the proposed token based data cleaning algorithms are explained clearly. Finally, the performance parameters such as i) Recall (RC) ii) false-positive error (FPE) iii) Reverse false-positive error (RFP) and threshold are used and make an analysis of the performance of the algorithms.

In this research, there are two points to be considered from the above paper. They are formation of tokens and maintaining log table for further steps of data cleaning. The selection of field for token construction is addressed in this research work. Formed tokens and similarity threshold values are maintained in LOG table for record similarity computation and duplicate data detection and elimination.

Erhard Rahm and Hong Hai Do’s paper on “Data Cleaning: Problems and Current Approaches”, classifies the state of the art in data quality problems that are addressed by the data cleaning [EH, 00]. Data cleaning is especially required when integrating heterogeneous data sources and should be addressed together with schema-related data transformations. Single-source and multi-source problems and schema and instance related problems are explained along with an example. Data cleaning involves several phases such as

i) Data analysis

ii) Transformation work flow and mapping rules

iii) Verification

iv) Transformation
v) Backflow of cleaned data

These phases are explained with examples. Finally, an overview of commercial data cleaning tools such as In data warehouses, data cleaning is a major part of the so-called ETL process Data analysis and reengineering tools, specialized cleaning tools and ETL tools are given in this paper.

The above paper contains details of single-source and multi-source problems. Unified schema for multi-source data warehouse and integrity constraints for the schema is addressed in this paper. Duplicate data detection and elimination is very big issue in multi-source data warehouse. Because data in integrated from multiple sources and maintained in data warehouse. Thus duplicate data should be purged out and complementing information should be consolidated and merged in order to achieve a consistent view of real world entities. In this research work, sequential based framework is proposed for duplicate data detection and elimination in order to reduce the time and improve the quality of the data.

Kanana Ezekiel and Farhi Marir’s paper on “Enhancing Data Preparation Processes Using Triggers for Active Data Warehousing”, examined that one of the problems of existing approaches is their limited support for data preparation for active and changing environments such as Active Data Warehouses [KF, 06]. Their focus is on static data preparation approaches. This paper addresses this limitation and a trigger mechanism designed to manage changes in a dynamic environment is utilized. The specification language of a trigger supports active and dynamic capabilities that enable
users to automatically filter or select and cleanse data at runtime. Background and related work of the data preparation process such as Information filtering, Integration, transformation and cleaning is discussed in this paper. The new approach uses triggers with dynamic capabilities to support data preparation tasks in active or dynamic environments. The new approach defines triggers with dynamic capabilities following the Dynamic Object Model (DOM) technology. The DOM technology is well placed to achieve the aim of flexibility. It will allow trigger and its components (Event, Condition, and Action) to be changes at runtime. This paper contains the details of the proposed approach such as trigger cleaning approach; trigger cleaning process and simple trigger cleaning examples are explained. TCA approach consists of two stages namely the trigger processing stage and the data processing stage. The main contribution of this paper is to enhance data preparation operations through triggers.

There are two points considered for this research from the above paper. They are triggers for active data warehouse construction and clustering algorithms to eliminate noisy data. Triggers are fired when preprocessed data is fed into the database. In this research work, software agent is used to perform the cleaning process and to maintain copy of each transaction in LOG file for the incremental data cleaning. Software agent will work according to the data. Clustering algorithm is proposed to eliminate duplicate data from the data warehouse.

Mauricio A. Hernández and Salvatore J. Stolfo’s, paper on “Real-World Data is Dirty: Data Cleansing and the Merge Purge Problem”, considers the data cleansing
of very large databases of information that need to be processed as quickly, efficiently, and accurately as possible [HS, 98]. Merging large databases acquired from different sources with heterogeneous representation of information has become an increasingly important and difficult problem for many organizations. Record linkage, the semantic integration problem or the instance identification problems are included in the merge purge problem. There are two issues: schema integration problem and heterogeneous representation of data and its implication when merging or joining multiple datasets considered as Merge purge problem. Sorted Neighborhood method is used for solving the merge problem. Sorted Neighbourhood method is used for solving the merge problem in three phases: Create keys, Soft data and merge. The effectiveness of the Sorted-neighbourhood method highly depends on the key selected to the sort the records. The selection of keys and corresponding example are explained. The comparison records, during the merge phase is determined by equational theory. An experimental merge purge algorithm is specified for new increments of data. Incremental merge purge algorithm is specified for new increments of data. It uses the multi-pass sorted-neighborhood method. Finally, initial experimental results on the incremental algorithm are specified based on the accuracy. The system provides a rule programming module that is easy to program and quite good at finding duplicates especially in an environment with massive amounts of data. This paper reports on the successful implementation for a “real-world” database that conclusively validates our results previously achieved for statistically generated data.
In the above paper, data cleaning takes place in two different places namely cleaning in merged data and incremental data cleaning [HGM+, 06]. This paper mainly deals with duplicate data cleaning. Separate file is maintained to store all the cleaning process for incremental data cleaning. There are two points considered for this research work. They are merging the new data with the existing data warehouse and fitting a table structure to a data warehouse. Incremental cleaning is very important because mainly data is duplicated while adding new data to the data warehouse. So, in this research work, selected attribute formed tokens, similarity threshold value, certainty factor value for each attribute are stored as LOG table. LOG table is very much important in incremental data cleaning for reducing the time.

Hui Xiong et al paper on, “Enhancing Data Analysis with Noise Removal”, focuses on removing noise and explores four techniques for noise removal to enhance data analysis. The techniques are

i) distance based

ii) cluster based

iii) an approach based on a Local outlier factor

iv) Hyperclique based data cleaner

The performance of the above techniques is evaluated. HCleaner leads to high performance as compared to outlier based data cleaning. The data cleaning techniques can be applied in two stages that is i) Data collection stage ii) Data analysis stage. SNM and AJAX can be applied in the data analysis stage. HCleaner uses the hyperclique
patterns as a filter to eliminate data object that are not tightly connected to other data objects in the data set. A hyperclique pattern follows the association pattern. Recall and False-positive errors factors are used to validate the data cleaning techniques. Finally, experiment results are also given in this paper. The experimental results show that all these methods can provide better clustering performance and higher quality association patterns as the amount of noise being removed increases, although HCleaner generally leads to better clustering performance and higher quality associations than the other three methods for binary data.

In the above paper, outliers are calculated using the clustering algorithm. The same way, clustering concept is considered and used in this research work for data cleaning to minimize the time taken for record comparison.