Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2024) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering (ICMLDE 2023)

# Leveraging Machine Learning For Enhanced Database Integration

Neha Reddy Palnati ,Vijay Kumar Reddy Julakanti, Nikhil Bayyavarapu

*Vellore Institute of Technology, Katpadi, Vellore 632014, India*

*Indian Institute of Technology Hyderabad, Kandi, Hyderabad 502285, India*

*Indian Institute of Technology Hyderabad, Kandi, Hyderabad 502285, India*

**Abstract**

In this research endeavor, our primary objective revolves around the intricate task of matching schemas across diverse databases. The challenge lies in recognizing the inherent relationships between attributes and employing sophisticated machine learning algorithms as our guide. Doing so aims to identify attributes that harbor comparable or akin values, thus facilitating potential mappings between the two databases for seamless integration. Our investigation delves into the complexities of mapping scenarios, encompassing the conventional one-to-one mappings and the intricate scenarios where a single attribute can map to multiple attributes. Within the scope of this report, an in-depth exploration of one-to-one mapping utilizing the powerful Kohonen Self-Organizing Maps is presented. The theoretical underpinnings of this approach and the experiments conducted to evaluate its efficacy are also elucidated. The outcomes and insights gained from these experiments are meticulously documented and presented. By unraveling the intricate nature of schema matching using machine learning techniques, this research contributes to the ever-growing body of knowledge in this domain. Our findings shed light on the potential of employing advanced algorithms to tackle the challenges of schema matching, paving the way for enhanced database integration.

*Keywords:* Schema Alignment; Artificial Intelligence; Kohonen Network; Levenshtein Distance; One to Several Mapping; Pairwise Mapping

## 1. Introduction

A database has a skeleton called a schema that shows the data [4]. The schema has the names of each thing in the database and its type. Sometimes, there is a need to link different parts of the database or move data between different databases. This means ensuring all the schemas match up so the data can be moved correctly. Schema matching is the process of matching up two sets of data represented by databases $X(x_1, x_2, x_3)$ and $Y(y_1, y_2, y_3)$ with $x_n$ and $y_n$ [11]. When both the datas are matched, the lookout is for attributes with similar names or meaning the same thing. The goal is to create the best match, ideally matching all the attributes from both databases. Consider the Tables 1 and 2. Here, the perfect schema mappings would be: *FName* + *LName* = *Name*, *Major* = *Maj_Stream* and *Address* = *House No* + *St Name*.

Table 1. Students

| FName | LName | SSN | Major | Address |
|---|---|---|---|---|
| Shruti | Jadon | 123-aaa-a | Computer Science | 1xx Brit Mnr |
| Ankita | Mehta | 234-bbb-b | Mathematics | 2xx Boulders |
| Tanvi | Sahay | 456-ccc-c | Political Science | 3xx N Pleasant St |

Table 2. Grad-Students

| Name | ID | Maj_Stream | House No | St name |
|---|---|---|---|---|
| Shruti Jadon | 123aaa | CompSci | 1xx | Brit Mnr |
| Ankita Mehta | 23bbb4 | Math and Stats | 2xx | Boulders |
| Tanvi Sahay | 45cccc | PoliSci | 3xx | N Pleasant St |

Researchers have had trouble automating the process of matching different sets of data [12]. This is because the data is created and named by humans and is specific to a particular type of information. So, people still need to help with the process to ensure the data is matched correctly. This takes a lot of work and time [10]. When an attempt to match two sets of data is made, the approach does not have to bother a person who knows about the subject all the time. This is where automated schema matching comes in [5]. There are two main ways to do this: In Table 1, an attribute has the potential to correspond with either one or multiple attributes in Table 2. Specifically, the pair (‘Major,’ ‘Maj_Stream’) signifies a one-to-one match between a single item in the first set and a corresponding item in the second set. Meanwhile, the pair (‘Address,’ [‘House No,’‘ St Name’]) indicates that a single item in the first set could match with multiple items in the second set. Advanced machine learning techniques can be used, and the structure of the schema to help us match up one-to-one items. However, when items are one-to-many, the need to get help from a person still arises. To match up items, data at each set can be looked at, or just the names of the items. Sometimes, there is a need to use both methods to get the best match.

Different ways of matching data sets with one-to-one matching are studied. A new way to match one-to-many data sets was devised, which hasn’t been done before. In the end, there were two ways of matching one-to-one data sets. Both of these methods use features to help match data sets. These features help to group similar data. The first method is called the centroid method. All the similar data from one set were grouped to use this method. Then, the data in the second set was compared with each data group in the first set. There are two methods to compare groups of data from two different datasets. The first method looks for the group in the first dataset that’s most similar to the data in the second dataset. The second method combines the attributes (characteristics) from both datasets and finds the most similar ones. In the initial technique, the centroid method, every attribute within the secondary collection is paired with at least one attribute in the primary compilation. More will be learned about this method in the upcoming sections. This new method being researched has two parts. The first part works pretty well. There’s a good chance of finding a match when one set of data is compared with another set of data. The second part is a little trickier. Sometimes, there won’t be a match between the two data sets. More discussion about this will happen later, alongside comparisons of all the different methods used to match data. There will also be a new way to match data sets that are more complex. It will explain how it works, and plausibly, it will be helpful because you won’t need a computer expert to help you match data.

In our collaborative research endeavor, we wish to underscore that both authors have been equally dedicated and have made substantial contributions across all aspects of this investigation. Beginning with the initial conceptualization of our research objectives, developing our innovative methodologies, meticulous data analysis, and creating novel data matching techniques, we have worked in tandem to drive this research forward. We want to emphasize the balanced and joint nature of our contributions, which reflects the collaborative spirit underpinning our collective efforts throughout this study.

The paper follows a structured layout across its sections, each contributing to the overarching goal. In Section 2, the approach is outlined, focusing on achieving one-to-one and one-to-many schema matches between the S and T data structures. Section 2.1 delves into schema data preparation, emphasizing the importance of standardized and labeled attributes in Table 3 and Table 4. Section 2.2 introduces the methodology for addressing one-to-many matches using a lexicon, while Section 2.3 details the process of achieving one-to-one schema matches through feature extraction and evaluation of the Centroid and Combined methods.

Moving on to Section 3, the paper discusses the experimentation and results obtained. It explores the utilization of clustering methods, including Self-Organizing Map (SOM) and K-Means Clustering, with different techniques and cluster sizes. Various distance measures are evaluated, and F1-scores are presented to gauge the effectiveness of these methods.

Section 4 serves as a closure, summarizing the paper's objective to enhance schema matching for limited data domains using a comprehensive glossary. It highlights the choice of the combined clustering method and the effectiveness of the edit distance technique for attribute matching.

Section 5 looks ahead to future work, suggesting potential directions for improving one-to-one mappings and expanding the mapping dictionary through semi-automated approaches such as web scraping and natural language processing.

## 2. Proposed Methodology

Finding one-to-one matches. In the rest of our explanations, let's say there is a data structure called S and another one called T. The goal is to find ways to match the information in T to the right information in S.

### *2.1. Schema Data*

Some information from a website called medicare.gov is being utilized. Only part of the information is used, though. Two lists of information are selected from the website that talks about how good different hospitals are at taking care of people for mental health reasons. One of the lists has 85 things to look at for each hospital in the United States. There are 1644 hospitals on the list in total. The"Provider_number" thing is used to make sure that there are no duplicates. The system is designed to find one-to-one and one-to-many matches between structures. The matching problem is first converted into a search problem to achieve this. Next, the system explores all possible ways that structures could match. The state's name is used to make sure the information is unique. This list has 74 things to look at and only has 52 different places because some states share information. Two new lists were curated using some of the information from each list. They were named Table 3 and Table 4, respectively. The first list is being used for experiments and is called "source schema." The other schema is referred to as "the test schema."

Table 3. IPFQR data - General

| tr_provider_number | tr_state | tr_hbips_2_overall_num | tr_hie_response |
|---|---|---|---|
| 10101 | AL | 23.7 | Yes |
| 40014 | AR | 1.47 | No |
| 34023 | AZ | 0.68 | Yes |

Table 4. IPFQR data - Statewise

| ts_state | ts_s_hbips_2_overall_num | ts_s_hie_yes_count | ts_start_date |
|---|---|---|---|
| AL | 2891.1 | 17 | 01/01/2015 |
| AR | 844.77 | 10 | 01/01/2015 |
| AZ | 4981.36 | 14 | 01/01/2015 |

This limited sample shows that the attributes pertain only to the domain and do not provide any details about the actual data content. As a result, methods that use attribute semantics for matching cannot be utilized.

A solitary database contains both data tables, which can be obtained using Python and Postgres. To ensure consistency between schemas, the attributes are prepared beforehand [2]. This cleaning process involves replacing any symbols in schema names, such as %, with their corresponding word form, i.e., *percent*, before schema matching is performed. Specific columns with integer or float data types might include character values, for instance, 'Not Available,' which are altered to 0. Additionally, characters like % found within the data are removed. The schema names are standardized to improve clarity and understanding by converting them to lowercase letters. In addition, when saving the data to the database, the source schema attributes are labeled with the prefix *'tr_'* while the test schema attributes receive the prefix *'ts_'*. This labeling helps distinguish which attributes belong to which schema and table.

## *2.2. One To Multiple Schema Mapping*

If a piece of information in a test matches with more than one piece of information in the source, the computer compares them in a one-to-many way. Sometimes, a person's address is written in one column or split into two columns labeled House Number and Street Name. To make the one-to-many comparison work, there is a need to make a list of all the ways an attribute can be linked to other attributes. This list is called a lexicon, and it helps the computer identify possible matches. The lexicon has keys and values. The "keys" are like labels that can be divided into smaller pieces, and the "values" are groups of labels related to each key. For example, the 'Address' key has 'House Number' and 'Street Name' values. If the source schema has a key that matches a value in the test schema, it counts as a one-to-many match. Any attributes that were already matched one-to-many way are not used when comparing the schemas one-to-one.

A big list is made that includes different sets of labeled information from around the world:
*Key*: Place, Site, Position, Spot, Dwelling
*Value*: Street Name, S_Name, St_Name, Str_Name, Stree_Name, StName, St_No, ST_Number, Street_No, S_No, S_Number, Street Number, StNumber, StNo, Apt_Num, Apartment_Number, Apartment Number, Apartment No, Apt_Number, Apt_No
*Key*: Moniker, Title, Appellation, Label, Identity
*Value*: Given Name, Given_Name, Forename, Personal_Name, Family_Name, Surname, SName, L_Name

The global lexicon is like a big book of words that shows all the different ways to say the same thing. This helps to match things together better. Right now, there are only two ways to match different things, but more ways can be added later by adding more examples to the book.

## *2.3. Matching Schema: Achieving One-to-One Mapping*

Once all one-to-many mappings have been identified, the next step was performing one-to-one schema matches on the remaining attributes. During this process, relevant descriptive elements are extracted from each attribute and leveraged to locate attributes with similar features across the two schemas under examination. Two methods are explored to extract these features - the Centroid and Combined methods. To evaluate the effectiveness of these methods, the F1-Score, which takes both recall and precision [13], is considered a criterion. Below is a detailed overview of each stage involved in one-to-one schema matching.

### ***Characteristics Crafting and Extraction***

When comparing attributes, the term 'features' pertains to the distinctive traits that provide an adequate and precise account of an attribute, enabling a side-by-side assessment with other attributes. By providing a measure of similarity or dissimilarity between attributes, these features are essential for attribute discrimination. The discriminators can be rooted in the attribute's data type or other aspects such as its range, domain, or the space it occupies, and these properties can be used to differentiate between the attributes. To enhance the comparison of attributes, features can be obtained from the schema specifications, such as the attribute's key status. These features can take binary values of 0 or 1 or be normalized to fall within the range of [0, 1]. Representing attributes with real values offers several advantages. Firstly, it allows for mathematical operations on the attributes, which were impossible with text, making it easier to measure their similarity. Secondly, features can be manually crafted to highlight the important aspects of an attribute when comparing it with others. 20 different 'discriminators' from the features provided by [6] to get information about our attribute set are used. The set of features is incomplete, but it can be made bigger to fit different types of databases or sources of information.

*Schema-Based Features* - By including schema information, the aim is to derive the subsequent characteristics: category of information, covering (Decimal, Integer, Character, Logical, date, time); measurement of fields indicated by the user; symbol key situation, identified as 1 for a key symbol and 0 otherwise; distinctiveness condition, noted as 1 for a symbol with a unique condition and 0 if not; and Non-Null restriction, presented as 1 for symbols with the restriction and 0 otherwise. All this information is obtained from the user-specified schema. To maintain homogeneity with symbol key situation, distinctiveness, and Non-Null restriction, the UNIQUE condition is solely employed when a symbol is not a key but remains singular. Every value is transformed into a genuine digit to maintain uniformity. Concerning categories of information, decimal sorts are presented with the digit 0, whereas time sorts are granted the digit 5. Field lengths are indicated with a numeric value. The remaining three attributes (i.e., symbol key situation, distinctiveness, and Non-Null restraint) are designated 1 or 0, indicating whether or not a specific symbol possesses this property. For example, a key attribute is represented as 1, while a non-key attribute is represented as 0. *Field-Based Features* - Numerous distinctive features can be extracted from data contained within a column. To differentiate between the various types of data, the data is categorized into three distinct groups: INT or FLOAT (age, salary), CHAR (name, city), and VARCHAR (address). These groups can include both numeric and character data. Then, features are generated to distinguish among these three data types, owing to the presence of specific information that might be unique to either numerical or VARCHAR attributes.

1. For a specific column, the following statistical measures can be extracted:

   - Mean - This represents the mean value of all entries in the given column.
   - Dispersion - This quantifies how much the values in the column deviate from their mean.
   - Relative Standard Deviation - This metric characterizes the variation of values regardless of the variable’s measurement unit.
   - Lowest - This denotes the minimum value present in the given column.
   - Maximum - This is the largest value in the column.

2. The character data features encompass:

   - Ratio of whitespace characters to overall length - This computes the proportion of spaces and other whitespace characters to text characters.
   - Ratio of non-alphanumeric characters (-’, _’, (’, )’, ‘\’ and so on) to total text length - This calculates the proportion of special characters to the total number of characters in the text.
   - Ratio of numeric characters to overall character council - This determines the ratio of all numeric characters to the overall number of characters in the text.
   - Proportion of plaintext characters to total characters - This assesses the proportion of ordinary text characters to the total count of characters.
   - Ratio of backslash characters to all characters - This measures the ratio of backslash characters to all the characters in the text.
   - Proportion of bracket characters to total characters - This quantifies the ratio of bracket characters to all the characters in the text.
   - Proportion of hyphen characters to total characters - This gauges the ratio of hyphen characters to all the characters in the text.

3. Shared Features for Numeric and Varchar Attributes
   For a specific column, the following statistical measures can be derived related to the length of the attribute used:

   - Mean Used Span - This denotes the typical length of the utilized attribute, as opposed to the aggregate length prescribed for all values within the column.
   - Spread of Used Span - This measures the deviation of the spans used by values in the column from their mean.

- Relative Standard Deviation of Used Span - This metric quantifies the variation in length of used attribute values regardless of the unit of measurement for the attribute.

A vector containing 20 items is created for each attribute to assess the attributes of both the source and test schemas. The said vector includes values associated with all 20 previously mentioned features. For instance, in the case of a character attribute, it may contain values pertaining to ratios concerning whitespace, special characters, brackets, and more.

ts_state (CHAR(2) PRIMARY KEY)

The feature vector for the test table in Table 4 has been provided in Table 5. Note that these features are not yet normalized and, therefore, do not conform to the 0-1 range.

Table 5. Features of attribute 'ts_state'

| | | | |
|---|---|---|---|
| Type of Data | 2 | Coeff of variance | 0.0 |
| Length | 2 | Minimum | 0.0 |
| Key | 1 | Maximum | 0.0 |
| Unique | 0 | No of Whitespace | 0.0 |
| Not Null | 1 | No of Special Char | 0.0 |
| Average Length Used | 1.0 | Ratio of numerics | 0.0 |
| Variance of Length | 0.0 | Ratio of chars | 1.0 |
| Var Coeff of Length | 0.0 | No of backslash | 0.0 |
| Average | 0.0 | No of brackets | 0.0 |
| Variance | 0.0 | No of hyphens | 0.0 |

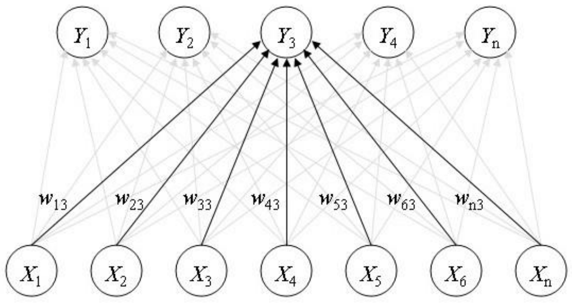
Once all the information is gathered for each attribute, the techniques explained in the sections are executed that follow to group attributes with similar features.

***Centroid Method for Clustering and Linguistic Matching***

To streamline the task of discovering matching symbols in the test schema, the initial method is utilized, which groups related symbols together in the origin schema. To realize this goal, an experiment with two clustering methods is conducted: the Kohonen Self-Organizing Map and K-Means Clustering. The upcoming sections furnish a thorough exposition of both techniques:

*Self Organising Map[3]:* The Kohonen Self-Organizing Map (SOM) is a type of computer system that can identify patterns in a large amount of data on its own, without being told what to look for (called unsupervised learning) [9]. The framework consists of two strata: an entrance stratum containing several nodes and an exit stratum containing additional nodes. Each departure node connects to all entrance nodes using distinct connections with attached magnitudes. When the network looks at a feature from the input, one output node will become active, and the network will try to make the weights on that node so it's similar to the input. This makes similar input features end up near each other. Using Kohonen SOM can help find similar attributes in a database, making it easier to search for attributes in the future, and even help with data analysis by identifying connections between different attributes. When the Kohonen SOM is presented with a characteristic of the input, the neuron with the weight vector that's the most similar to the input is chosen and identified as a category. The weight values of all the neurons touching this neuron (and itself) are adjusted to be more similar to the input vector. This is done many times to make the network better (learn) at categorizing future inputs. To cluster data, a 20-feature vector with as many examples as there are characteristics in the source is input into Kohonen SOM. You can determine the maximum number of clusters depending on the number of output neurons. You can either select the number of clusters yourself or have the computer decide for you.

*K-Means Clustering: An Overview -* K-Means Clustering is a repeating process that joins similar data points by working in a D space that contains N data cases [1]. The algorithm relies on two different interchanged steps until the results stabilize. In the first step, the algorithm picks k cluster center points scattered around D in a random manner. In the current scenario, N data points with a size of 20 vectors are fed as input to the K-Means algorithm, which includes the same number of data points as there are characteristics in the source database. Here's what happens in the next step: a) Each data point, denoted as $x_i$, is compared to the nearest centroid (group) using Euclidean distance. The point is



then added to that group. b) After the groups are formed in the initial step, the algorithm examines the groups' average to generate new centroid points. This lets the algorithm start again at step one. The K-Means Clustering approach is handy for grouping things in source schema databases and limiting the amount of trial and error needed when looking through the schema of a test.

The chosen way to group the data breaks the data into 7 different groups. To decide which grouping is best, the silhouette score calculation is used. This measures how much a piece of data matches the group it belongs to in comparison to other groups. On a scale of -1 to 1, the silhouette score tells you how good the grouping method is, with scores nearer to 1 being better.

Once the clusters have been created, the algorithm proceeds to find the centroid of every single cluster. This centroid value represents the average feature vector of all the items present in that cluster. As a final step, the algorithm measures the Euclidean distance between every cluster centroid and each test attribute, $t_s$, to determine which source attributes each test attribute should be associated with. The algorithm first calculates the distance between the test attribute $t_s$ and each cluster centroid. The cluster whose centroid is closest to $t_s$ is then identified using this distance metric. The algorithm restricts the comparison of $t_s$ to only the candidate match space within this particular cluster of source attributes. This step helps to limit the search space and improve the speed of the matching process significantly. Once a matching space is assigned to $t_s$, the algorithm performs a linguistic mapping of every candidate match within the space back to $t_s$. Subsequently, the algorithm assigns a similarity score to each candidate match to indicate the extent of similarity it shares with $t_s$.

To match attribute names, an approach called linguistic matching [15] is adopted. This technique leverages methods from natural language processing to measure similarity or dissimilarity between two attribute names using an edit distance process. In the edit distance method, the smallest number of actions needed to change one string into another are counted [7]. This method is used for each test-source attribute pair to identify the likelihood of a match by measuring the similarity between the two attribute names. The more similar the attribute names are, the higher the probability of a match between them.

## *Enhancing Schema Matching using Clustering and Linguistic Analysis*

The combined approach of clustering and linguistic matching involves consolidating the attributes from the source and test schemas and clustering them based on their similarities. The clustering algorithm is furnished with a characteristic list of 20 elements as its input, identical in number to the collection of entries contained within both schemas. Then, the attributes are divided by their source and test group. For each test attribute in a cluster, the edit distance method is used to measure its linguistic similarity with all the source attributes in the same cluster. This process helps us infer probabilities of matching for each source-test attribute pair. However, it's important to note that linguistic matching doesn't apply to clusters that contain only source or test attributes.

## *Evaluation of One-to-One Mappings Using F1-Score*

In this study, the efficacy of the Centroid and Combined methods in attribute matching is assessed, using the F1-score as a metric to evaluate their performance. The one-to-one matches found from these methods are evaluated by analyzing how well they balance precision and recall. Then, they are compared to the actual attribute matches. To measure how accurate the clustering evaluations are, a method called F1-score is used. It calculates an accuracy score by weighing precision and recall together. Accuracy evaluates the proportion of correct positive results compared to the total number of positive results. Completeness evaluates the proportion of accurate positive results compared to the overall number of factual positive results. Using the F1 measure, a strong assessment of the proficiency of our techniques in matching attributes can be derived.

Manual mappings are used as a basis for our evaluation to calculate the values for precision and recall. Using these mappings, the number of true positives is determined- namely, the accurate matches found by our method - as well as the number of false positives, where our method returns a match that is incorrect, and false negatives, where our method fails to return a match that should have been identified. This helps us determine the accuracy of our matching method and identify areas where improvements are needed. The F1 score is then defined based on these values:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1}$$

Where

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3}$$

The proposed methodology introduces several key improvements compared to traditional methods:

Comprehensive Lexicon: The use of a lexicon to manage one-to-many matches provides a systematic and well-defined approach to handle complex data associations.

Clustering and Linguistic Analysis: The combination of clustering techniques and linguistic analysis enhances the matching process, making it more robust and efficient.

Feature-Based Matching: The paper leverages schema-based and field-based features to assess attribute similarity, adding a layer of precision to the matching process.

Automated Matching: By employing clustering and linguistic techniques, the methodology significantly reduces the need for manual intervention in schema matching, saving time and effort.

Evaluation Metrics: The use of the F1-Score as an evaluation metric provides a quantitative assessment of the methodology's performance, allowing for clear comparisons and improvements.

## 3. Experimentation and Results

In our experimentation and analysis, the source and test schemas that were previously defined are used. Various methods were explored at each step, including the selection and extension of different sets of features, which were found to be dataset-dependent. In our specific dataset, it was observed that incorporating data on special characters was essential for improving cluster accuracy. Following selecting the standard feature set outlined in [6], dataset-specific characteristics were integrated to refine the clusters, as verified through manual observation.

Our dataset is initially subjected to one-to-many mapping with the help of the dictionary created as detailed in the previous section. With this method, the successful identification of the sole mapping present in our database is viable. Next were the experiments with one-to-one mapping.

To begin our experimentation, the focus was on clustering methods. Specifically, SOM and K-Means clustering were tested using two techniques, centroid and combined, over a range of 7 clusters: 25, 30, 35, 40, 45, 50, and 55. Although selecting the cluster count depends on the user's discretion, a range of 7 values that balance narrowness and wideness was chosen. To examine the findings of our experiments, consult Figures 1, 2. These figures showcase the silhouette scores of each cluster size; one is displayed using the centroid method, and the other using the combined method. Please ensure that the references to the figures remain unchanged.

After contrasting the outcomes, the conclusion was that SOM exhibits marginally superior performance compared to K-Means with the adopted techniques. The optimal scores were reached using the centroid approach with 40 clusters and the merged procedure with 50 clusters. Therefore, the decision to utilize SOM for clustering the data was made, selecting 40 clusters for the centroid approach and 50 clusters for the combined approach.

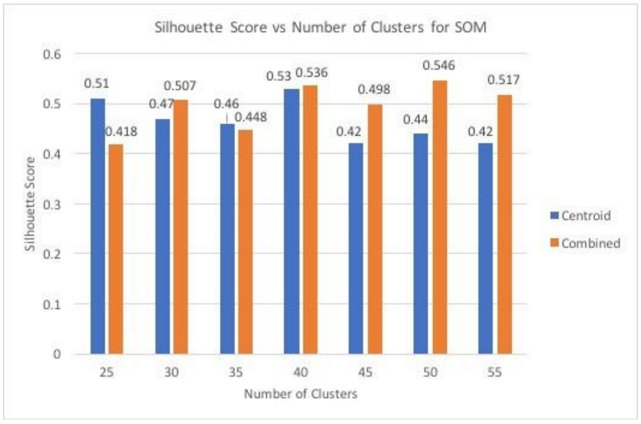


Fig. 1. Silhouette scores versus number of clusters for SOM

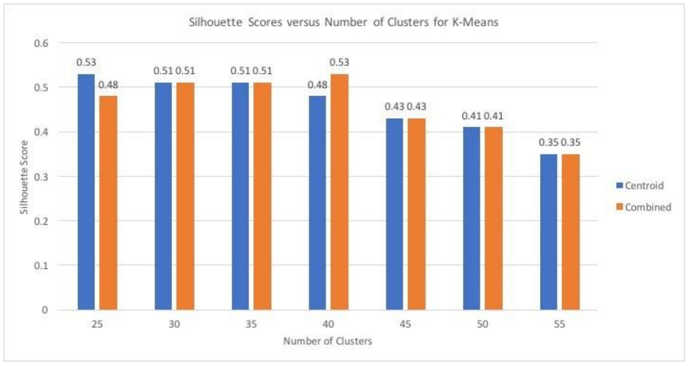


Fig. 2. Silhouette scores versus number of clusters for K-Means

Two other methods besides edit distance were used to match attributes within clusters. One of these methods is called "Euclidean Distance." It works by finding the difference between the features of two things being compared. This difference can be calculated with math like this:

$$\sqrt{(s_1 - t_1)^2 + (s_2 - t_2)^2 + ... + (s_n - t_n)^2} \tag{4}$$

This refers to two feature vectors, one for a source attribute and one for a test attribute. The $i^{th}$ feature in each vector is represented by $s_i$ and $t_i$, respectively. If $s$ and $t$ are utilized to represent the source and test vectors, then cosine similarity will measure the cosine angle between them to determine their similarity.

$$\frac{\sum_{i=1}^{n} s_i t_i}{\sqrt{\sum_{i=1}^{n} s_i^2}\sqrt{\sum_{i=1}^{n} t_i^2}} \tag{5}$$

To assess the performance of various distance measures, the F1-scores were calculated using all three measures for two distinct clustering methods: centroid and combined. The F1-scores for the centroid method were obtained for the cluster size of 40, while for the combined method, they were analyzed for a cluster size of 50. Check out Tables 6 and 7 to observe the findings of these assessments for the centroid and merged procedures, correspondingly.

As per the earlier results, it appears that using the edit distance approach is superior in creating precise one-to-one links between source and test schema attributes. Furthermore, the merged clustering technique produces superior outcomes compared to the centroid method, exhibiting the maximum F1-measure of 0.71.

In contrast to the outcomes reported in [8], our method is not as efficient, as the corpus-dependent procedure attained the highest mean F1-score of 0.87. Nonetheless, as expounded in [6], the IBM AS/400 database displayed a

Table 6. F1-scores for centroid method

| Distance Measure | Precision | Recall | F1 score |
|---|---|---|---|
| **Edit Distance** | **0.527** | **1.0** | **0.690** |
| Euclidean Distance | 0.222 | 1.0 | 0.363 |
| Cosine Similarity | 0.0555 | 1.0 | 0.105 |

Table 7. F1-scores for combined method

| Distance Measure | Precision | Recall | F1 score |
|---|---|---|---|
| **Edit Distance** | **0.54** | **1.0** | **0.71** |
| Euclidean Distance | 0.194 | 1.0 | 0.325 |
| Cosine Similarity | 0.027 | 1.0 | 0.054 |

maximum similarity of 0.995 regarding output match correlation. By employing our technique, a peak similarity of 0.984 was achieved for the IPFQR dataset using the edit distance and centroid procedure. Despite our findings differing from the mainstream approaches, dependable one-to-one correlations with a moderate amount of information were obtained.

## 4. Closure

The goal of this project is to address the challenges of performing one-to-many and one-to-one schema matching with limited data. To achieve this, we proposed generating a comprehensive glossary which encompasses all probable one-to-many linkages that can be updated with time to enhance the one-to-many matching procedure. This technique is especially advantageous for challenges with limited domains, where only a restricted number of one-to-many connections are plausible, and it also provides users with the opportunity to personalize the associations. Whenever a user incorporates a fresh attribute that may suit a one-to-many linking, they can append it to the dictionary for others to benefit. We came up with a method that centers on making a model that groups together similar attributes from the original schema when trying to match them one-to-one. By capitalizing on both the schema titles and the tabular data, we formulated an approach that curtails the quest for corresponding attributes for each trial attribute, rendering a precise one-to-one linkage.

We tested two methods we came up with, and found that the combination method works better for our dataset than the centroid method. We posit that this is attributable to the fact that the merged approach enables a test attribute to not align with any trained attribute, a possibility that the centroid procedure doesn't afford. Moreover, we observed that for matching comparable attribute names, the edit distance technique outperforms the cosine or Euclidean distance procedures by a wide margin. This is particularly evident in our dataset where several analogous attributes possess analogous names. If attribute names denoting the same thing differ significantly, the edit distance technique may not perform as adeptly as the other two similarity measures. Typically, while comparing schemas of tables that belong to the same database, comparable or indistinguishable names are probably employed to denote similar entities, thereby rendering edit distance as the favored approach for measuring similarity. However, when the tables originate from disparate databases, the two schemas are prone to possessing dissimilar names, which makes edit distance an unsuitable measure of similarity.

Overall, although the proposed methods may not yield superior performance compared to existing ones, their low data requirements make them practical tools for implementation, particularly in systems that can leverage attribute name similarity.

## 5. Future Work

Currently, the one-to-one mappings employed are rather elementary and lack the capability to facilitate complex mappings. To enhance the feasibility of such mappings, a cue can be taken from [14] and devise search algorithms capable of accommodating more intricate mappings. The creation of the mapping dictionary could also be refined to incorporate semi-automated procedures rather than relying solely on manual input. One method involves using web scraping to detect frequently encountered one-to-many connections and adding them directly to the glossary. An alternative is to employ natural language processing assignments such as forming clusters of different-sized words to produce an exhaustive compilation of all feasible word clumps to incorporate in the one-to-multiple correlation reference book.

## References

[1] Ahmed, M., Seraj, R., Islam, S.M.S.: The k-means algorithm: A comprehensive survey and performance evaluation. Electronics **9**(8), 1295 (2020)

[2] Ghannad, P., Lee, Y.C., Dimyadi, J., Solihin, W.: Automated bim data validation integrating open-standard schema with visual programming language. Advanced Engineering Informatics **40**, 14–28 (2019)

[3] Kohonen, T.: Self-Organizing Maps. Springer (1997)

[4] Koupil, P., Hricko, S., Holubova´, I.: Schema inference for multi-model data. In: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems. pp. 13–23 (2022)

[5] Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A., Katsifodimos, A.: Valentine: Evaluating matching techniques for dataset discovery. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE). pp. 468–479. IEEE (2021)

[6] Li, W.S., Clifton, C.: Semantic integration in heterogeneous databases using neural networks. Proceedings of the 20th International Conference on Very Large Data Bases **VLDB '94**, 1–12 (1994)

[7] List, J.M.: Beyond edit distances: Comparing linguistic reconstruction systems. Theoretical Linguistics **45**(3-4), 247–258 (2019)

[8] Madhavan, J., Bernstein, P.A., Doan, A., Halevy, A.: Corpus-based schema matching. Proceedings of the 21st International Conference on Data Engineering **ICDE '05**, 57–68 (2005)

[9] Murugesan, V.P., Murugesan, P.: Some measures to impact on the performance of kohonen self-organizing map. Multimedia Tools and Applications **80**(17), 26381–26409 (2021)

[10] Peng, X., Liu, H., Siggers, K., Liu, Z.: Automated box data matching for multi-modal magnetic flux leakage inspection of pipelines. IEEE Transactions on Magnetics **57**(5), 1–10 (2021)

[11] Rahm, E., Peukert, E.: Large-scale schema matching. (2019)

[12] Sahay, T., Mehta, A., Jadon, S.: Schema matching using machine learning. In: 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN). pp. 359–366. IEEE (2020)

[13] Yacouby, R., Axman, D.: Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In: Proceedings of the first workshop on evaluation and comparison of NLP systems. pp. 79–91 (2020)

[14] Ying, Q., Liwen, Y., Zhenglin, L.: Discovering complex matches between database schemas. 2008 27th Chinese Control Conference pp. 663–667 (2008)

[15] Zhang, Y., Floratou, A., Cahoon, J., Krishnan, S., Mu¨ller, A.C., Banda, D., Psallidas, F., Patel, J.M.: Schema matching using pre-trained language models. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE). pp. 1558–1571. IEEE (2023)