Schema Matching using Machine Learning

Ankita Mehta
amehta@cs.umass.edu

Shruti Jadon
sjadon@cs.umass.edu

Tanvi Sahay
tsahay@cs.umass.edu

***Abstract*—In this project, we deal with the problem of matching schema of different tables across databases with each other with the help of certain machine learning algorithms. This is done in order to recognize which attributes contain the same or similar values and might map to each other in case the two databases are to be used in unison. We also tackle the challenge of a single attribute mapping to multiple attributes along with the case of basic one-to-one mapping. For this report, one to one mapping using Kohonen Self-Organizing Maps has been explained and experiments carried out have been presented.**

I. Introduction

Schema matching is one of the key stepping stones for performing data integration and automating this task has been a topic of research for several years. In simpe terms, schema matching can be explained as follows: Given two databases $X(x_1, x_2, x_3)$ and $Y(y_1, y_2, y_3)$ with x_n and y_n representing their attributes resepectively, we match a schema attribute to another either if it is semantically similar or if it represents the same data. Consider the Tables I and II. Here, Schema Matching would match SSN with ID (both unique identifiers) and Major with Maj_Stream (both Student Majors).

TABLE I. Students

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

TABLE II. 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

Intutively, there are several issues to be dealt with before schemas can be accurately matched with each other. Consider the examples above. The first two columns of Table I matches with the first column of Table II, while the last column of table I matches with the last two of table II. Similarly, basic semantic matching of attribute names cannot match terms like SSN and ID together. One method to overcome this can be creation of a dictionary to hold such mappings and simply query the dictionary everytime an entry like this occurs. However, even with such mappings, it is almost impossible to capture all such possible similarities. Also, since language itself is dynamic, maintaining a dictionary of this sort would be space as well as time intensive and will require constant manual supervision.

In the past, reseach has been done on automating schema matching between different tables using graph based as well neural network based approach. These methods have considered using the content of each column and other attribute features as the recognition metric in addition to the name of the attribute itself. A detailed explanation of these methods will be given in the next section. However, none of these methods solve the case of many to one or one to many attribute matching using machine learning techniques.

In this project, we propose a preliminary way of adding many to one mapping to the already present methods of one to one mapping using machine learning techniques such as self organizing maps and neural networks. This report covers the implementation of self-organizing maps to perform one to one matching and compares the performance with manual matching. Section 2 provides brief descriptions of the previous work while section 3 describes the features extracted for each attribute. Section 4 explains one to one schema matching while sections 5 explains the experimentation carried out. Finally, section 6 describes the results and observations followed by a description of future work to be carried out in section 7.

II. Related Work

In this section, we briefly present the previous work associated with Schema Matching and how it associates with our current approach.

A. *Semantic Integration in Heterogenous Databases using Neural Networks[4]*

This paper discusses a method of schema matching that employs neural networks in order to learn which subset of attributes of a database A can an attribute of the database B belong to and what their percentage of similarity is. Features of each column are extracted using a database specific parser and these features, represented as vectors with each value lying in the range (0,1) are used as identifiers for that column and they are clustered together using a self organizing map. Then cluster centers are calculated and a single hidden layer neural network with M outputs neurons (M = number of clusters) is trained with output as the similarity percentage of the attribute with each cluster.

While this method, known as SemaInt, provides the user with a similarity mapping of each attribute in one schema with a set of attributes in another, it does not take into account the fact one might map to a set of others as well.

B. *Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach[2]*

The authors in the paper have explicated the need of companies to uniformly access multiple sources of data together. This paper solves the schema matching problem using LSD (Learning Source Description) approach which is nothing but an extension of the Machine Learning classification problem.

It has two phases: Training Phase and Matching Phase. In the Training phase, firstly, the authors manually specify schema mappings (i.e. provide labels to the schema) for some sources and extract some of their data. Then they train the two base learners: Nave Bayes and Name Matcher using the extracted data and the labels assigned manually. After training, cross-validation approach is used and each learner is given a weight. In the matching phase, the match of each remaining data using the trained base learners is found and predictions from both learners are then combined into a single prediction using the weights calculated in the training phase. This whole approach is constrained only for one to one mappings. It differs from SemaInt in that authors are manually matching a subset of the schemasand using this information to train their systems, as opposed to extracting descriptive features for each attribute. One of the major drawbacks of this approach is again that the possibility of one attribute mapping to a set of attributes has not been considered. Also, a large dataset is required to perform the necessary training of the classifiers.

*C. **Database Schema Matching Using Machine Learning with Feature Selection**[5]*

This paper is discussing about a tool called Automatch for automating the schema matching process. It matches each attribute of one client to another client schema. It uses feature selection to determine what attributes of the global dictionary will be more relevant to a given schema. This approach consists of a dictionary which is created by using schema examples and tuned by domain experts. Dictionary includes various clusters of attributes say R1, R2, R3 etc. It compares attributes of one schema (S1, S2, S3etc) with each of the dictionary attributes (R1, R2, R3 etc) and assign a weight based on probability formula of symmetry. The same is repeated with another schema and a path from schema 1 to schema 2 via the dictionary is chosen. The Minimum Weight Path determines which attribute of schema 1 is closely aligned with which schema 2 attribute.

While this method improves on its predecessors by including one-to-one attribute matching rather than just matching one attribute with a set of possible attributes, it still has the same problem that it does not consider the possibility of one attribute matching to a set of attributes.

While all three of these papers present different methods of solving schema matching using a machine learning technique, none of them take into consideration the one-to-many matching scenario. Other papers, such as CUPID[1] solve this problem, however they do not make use of machine learning, which is why they have not been considered here.

III. Feature Extraction

For each attributes, 17 hand made features were extracted and used as their descriptors for all future training. The features were divided into three categories: specific to numeric values, specific to character values and common to both. The features extracted were:

Common to both

- Type: if Type is INT then set to 0 , Numeric to 1, Text 3, Varchar to 4 etc
- Length: Length of the datatype defined.
- Key: If the given attribute is a Primary Key or Foreign Key or not[0 for no 1 for yes]
- Unique: if the given attribute has Unique Property.
- Not Null: if the given attribute has constraint of Not Null.
- Average Used Length: It is summation of used length to total length allocated.
- Variance of Length: It is variance of Used Length array.
- Variance Coefficient: It is Coefficient of variance for Used Length array.

For Num Values

- Average: Average of Specific Attribute values
- Variance: Variance of Attribute Entries
- Coefficient of Variance: Coefficient of Variance of Attribute Entries
- Minimum: Minimum of Attribute values
- Maximum: Maximum of Attribute Values

For Text/Char Values

- Number to All: It calculates the Number of numeric values to Used Length
- Character to All: It calculates the Number of character values to used length
- Special Characters to All: It calculates the Number of Special Characters Values to Used Length
- White Space to All: It is ratio of the Number of Space to Used Length

For each attribute, all of these 17 features will be extracted and the resultant feature vector will act as the list of values that represent the particular attribute.

IV. One to One Schema Matching

For one to one schema matching, we applied the same methodology follwed by the authors of SemaInt. After extraction of features from each attribute, we prepared a self-organizing map to cluster attributes within the same database such that attributes containing similar data (such as SSN and Employee ID) will be clustered together. After preparation of this model, attributes from the test set were matched with the prepared clusters. Here, it has been assumed that each attribute in the test set will map to at least one attribute in the training set. A short summary of self-organizing maps has been given below:

A. Kohonen Self Organizing Maps [3]

A Kohonen Self-Organizing Map is a type of Artificial Neural Network which is trained using unsupervised learning method in such a way that similar patterns in the input data are clustered together. A general architecture of the map has been shown in Figure 1, with the input layer having N nodes and the output layer having M nodes. Each output neuron is connected with every neuron in the input layer and each connection has a weight associated with it. For each input feature, a single output neuron is fired such that the weight vector associated with this neuron is closest to that input vector. Weights of all neurons near this activated neuron, including its own, are updated in such a way that it brings them closer to the input feature vector. Over several iterations, these weights are learned by the network and for any new input, the neuron with the weight vector closest to it is chosen as its class.

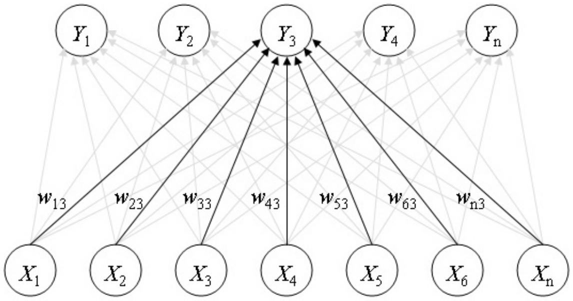

Fig. 1. Self-Organizing Map Architecture

In our implementation, the input provided to SOM was a size 17 feature vector with number of examples equal to number of tuples in the training database.

V. Experimentation

A. Dataset

Before explaining the experiments carried out, we describe the dataset used for the purpose of this project. The dataset is from the medical domain and contains tables for Hospital Comparisons. We have used two data tables for our project: One is the Hospital Level table, which contains data from several hospitals and the other is the State Level table, which contains a single hospital for every state in the US. Excerpts of both are given in Table III and Table IV. The first table contains 85 different attributes and 1645 instances of data while the second one has 74 attributes and 52 data tuples.

TABLE III. Hospital Level Table

Hospital_Name	Start_Date	TOB-1_%	Hie_Measure_Description
Dale Medical Center	01\01\2015	73.47 %	yes
Citizens BMC	01\01\2015	68.75 %	yes
Shoals Hospital	01\01\2015	67.24 %	no
St. Mary's Hospital	01\01\2015	30.58 %	yes

TABLE IV. State Level Table

State	Start_Date	S_TOB_1_Den	S_Hie_Yes_Count
AL	01\01\2015	1286	0
AR	01\01\2015	13981	17
CA	01\01\2015	7623	10
AZ	01\01\2015	46624	30

Models have been trained using Table III and tested over Table IV. As can be seen, Table III contains attributes that do not occur in Table IV and vice-versa. Certain attributes in the tables can be mapped directly to each other, such as $Start\_Date$ and $State$ while others such as $Hie\_Measure\_Description$ and $S\_Hie\_Yes\_Count$, which might semantically appear to be similar contain very different data and hence should not be matched with each other.Therefore, Semantic similarity is not sufficient for schema matching. More detailed description of the downloaded dataset can be found here: https://data.medicare.gov/data/hospital-compare.

B. Pre-processing

Before beginning the experimentation, data as well the schema was cleaned in order to allow proper processing of the tuples. The following cleaning steps were undertaken.

- Schema names were cleaned to convert symbols such as $\%$ to words like $Percent$ to allow uniformity across schema.
- Certain numeric data columns had values such as "Not Available" in them. All of these were converted to 0 to allow the data type to be numeric which in turn let us compute all statistical properties for these columns. Without changing this, the data type would have been VARCHAR, which would have resulted in loss of information.
- The $\%$ symbol was also removed from all numeric columns for the same reason given above.

C. Methodology

Once the data was cleaned, features for each column for both the training and testing tables were extracted.Then using Kohonen Self Organising Maps, similar features were clustered together. Examples of the extracted features and their cluster ID for training dataset can be seen in Table V.

TABLE V. Attributes, Feature Vectors and corresponding clusters

Attribute	Feature Vector	cluster id
hospital_name	2, 500, 0, 0, 0, 274517.6471, 0.1392, 0.2939, 0 0, 0, 0, 0, 0.1, 0.1057, 0.0001, 0.9999	5
start_date	4, 4, 0, 0, 0, 773.6471, 0.0, 0.0, 0, 0, 0, 0, 0 0.0, 0.2, 0.8, 0.2	5
TOB-1_%	2, 500, 0, 0, 0, 62764.7059, 0.0035, 0.2038, 0, 0, 0, 0, 0, 0.0054, 0.3027, 0.6321, 0.3679	2
Hie_Measure_Description	2, 500, 0, 0, 0, 502870.5882, 0.0, 0.0, 0, 0, 0, 0, 0, 0.0962, 0.0962, 0.0, 1.0	5

The Self Organizing Map provides spatial coordinates of the activated neuron as output for each attribute i.e. for an attribute "State", the SOM will give an output of (2,2) corresponding to the neuron at that position in the output grid. For ease of clustering, each grid point has been mapped to a unique cluster id as: (0,0) $->$ 0, (0,1) $->$ 1 and so on.

Once the model was trained, extracted feature vectors for the test set were classified into cluster using the following approaches:

- SOM - After clusters were prepared using SOM, the test feature vectors were provided as input to the network one by one and their correspoding outputs were noted.

- Cluster Center Matching - For each unique cluster, the center of that cluster was computed and distance of every test attribute from these cluster centers was found. The test attribute was assigned to that cluster whose center was closest to it.

VI. Results and Observations

Experiments were perofrmed on two cluster grid sizes: 5x3 and 7x7. For grid size 15, 10 clusters were observed, out of which 5 have been shown in figure 2. For grid size 49, 28 clusters were observed, out of which 24 clusters have been shown in figure 3. The values shown are for the same attributes for different grid sizes.

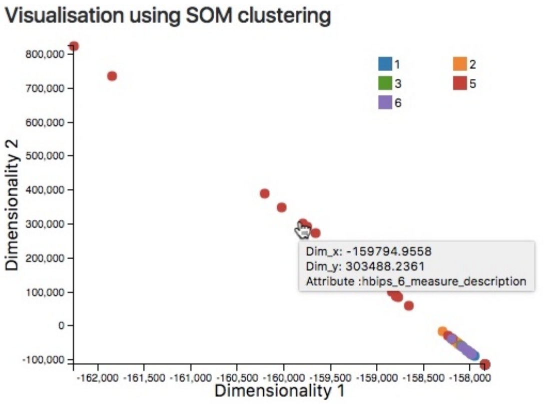

Fig. 2. Clusters for grid size 15

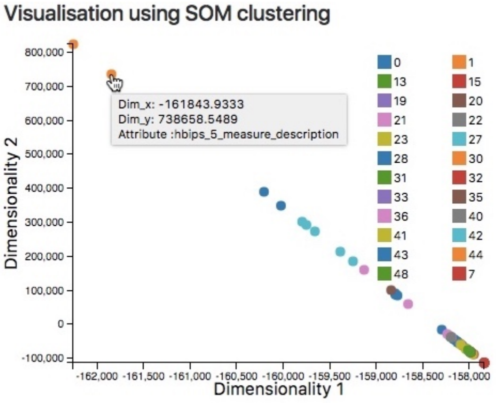

Fig. 3. Clusters for grid size 49

For grid size 15, table VI shows the clusters and attributes lying in each cluster, referred to by their attribute id, which corresponds to their position in the table.

For test table, clustered using method 2 provided in the methodology, a subset of expected and predicted clusters has been shown in table VII.

Observations

From the above results, it can be inferred that most of the attributes containing similar data have been clustered together.

TABLE VI. Cluster ID and underlying attributes

Cluster ID	Set of Attributes	Majority Type
1	4, 8, 11, 13, 16, 21, 26, 31, 36, 41, 45, 49, 51, 52, 55, 57, 58, 65 , 69, 76, 81	footnotes
2	18, 23, 28, 33, 38, 43, 46, 62, 66, 73	percentages
3	19, 63, 64, 67, 68	num/den
5	1, 2, 3, 6, 7, 12, 17, 22, 27, 37, 42, 50, 53, 54, 56, 59, 60, 70, 71, 72, 77, 82, 83	text data
6	9, 14, 20, 24, 25, 29, 30, 32, 34, 35, 39, 40, 44, 47, 48, 74, 75, 79, 80	num/den
7	78	percentage
8	61	long decsription
9	10, 15	den
12	5	zip code
13	0	provider no

TABLE VII. Predicted vs Expected outcome

Attribute	Predicted Cluster	Expected Cluster
s_flu_season_start_date	2	5
state	2	1
s_peoc_no_percentage	2	2
s_fuh_30_numerator	3	3

From figures 2 and 3 we can see that grid size 15 provides slightly more general results. As can be seen, for grid size 49, descriptions were divided into 6 clusters while in grid size 15, they were all included in a single cluster. Since we are going for more general results right now rather than the fine tuned ones, we decided to use the grid size 15 rather than any other value.

With the feature prepared right now, some of the attributes show similarity with others which do not contain related data. For example, at present, dates are being clustered together with descriptions, which is not desirable. To correct this, more features, namely: number of “ – ”s / Total Length and number of “ \ ”s / Total Length will be added. More features can also be added depending on the future performance of the model.

From the table VII shown above, it can be seen that test attribute clustering was unsatisfactory. The result obtained was same for both the techniques provided in methodology. For technique 1 i.e. clustering using SOM, output classes which were not observed in the training phase were also present in the testing phase and for technique 2 i.e. distance from cluster centers, a very low percentage of test attributes were being classified properly. To correct this, other methods of assigning classes, such as nearest neighbors and neural networks, will be tested.

VII. Limitations and Future Work

The work done till now has several limitations which we will try to overcome in the next phase of the project. Firstly, we have not yet considered matching a test attribute to a single train attribute. Instead, we are simply providing a cluster identity to the test input attribute which only tells us to which set of train attributes does our test attribute belong. Secondly, name of the attribute has not been taken into consideration yet, which may help in matching individual attributes with each other. Finally, we are yet to perform the one-to-many attribute matching.

The next task in the pipeline is to prepare a dictionary of possible one to many mappigs and recompute the feature vectors for each attribute accordingly. Once the new features have been calculated, the same procedure will be applied again to match a train and test database. Once each test attribute has been assigned to a cluster, the attribute will be matched with every train attribute in the corresponding cluster and one to one mapping will be assigned. For example, if a test attribute ID has been assigned to a cluster C that contains: $SSN, EmployeeID, ProductID$, this attribute will be individually matched with each of the values in C and the one that it is associated most closely with is returned as the final matching. Finally, we will try other clustering algorithms and try other methodologies and test how they perform with the same task to improve the accuracy of our system.

References

[1] Jacob Berlin and Amihai Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the 14th International Conference on Advanced Information Systems Engineering*, CAiSE '02, pages 452–466, London, UK, UK, 2002. Springer-Verlag.

[2] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, pages 509–520, New York, NY, USA, 2001. ACM.

[3] Teuvo Kohonen. *Self-Organizing Maps*. Springer, 1997.

[4] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 1–12, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[5] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 49–58, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.