

# Schema Matching using Machine Learning

Ankita Mehta

Shruti Jadon

Tanvi Sahay

**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 and Multilayer Perceptrons 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 automatic this task has been a topic of research for several years. In simple 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 respectively, 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

Intuitively, 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 match 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, research 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 the methodology and technique used for performing attribute matching and section 5 details the experimentation carried out and results obtained.

## 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

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

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 schemas and 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***

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 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:

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

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 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**

### ***A. Kohonen Self Organizing Maps***

Kohonen Self-Organizing Maps is a type of Artificial Neural Network which is trained using unsupervised learning method in such a way that similar patterns are clustered together in the input data. In this technique, when an input data activates a neuron then weight of activated neuron as well as the adjacent neurons are updated over the various iterations. Therefore, SOM forms a semantic map where similar samples are mapped closed to each other and dissimilar ones get apart. Figure 1 shows a self-organizing map network architecture with an input layer having N nodes and an output layer having M nodes. Here N is the size of feature vector of each input and M are the number of clusters to which that input data is mapped and  $W_{ij}$  is the weight vector assigned to the neurons over the iterations.

### ***B. Multilayer Perceptron***

This section only if we decide to do this finally.

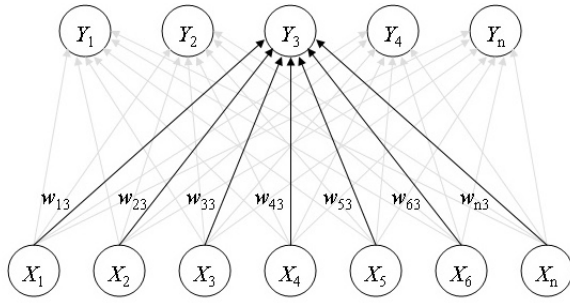


Fig. 1. Self-Organizing Map Architecture

## V. EXPERIMENTATION

### A. Dataset

### B. Data Labelling

We might need to hand label the attributes as a base line to check the accuracy of our system.

### C. Results

Clustering accuracy table, scatter plots etc.

## VI. LIMITATIONS

No many to one mapping and anything else you can think of.

## VII. FUTURE WORK

### REFERENCES

- [1] H. Kopka and P. W. Daly, *A Guide to L<sup>A</sup>T<sub>E</sub>X*, 3rd ed. Harlow, England: Addison-Wesley, 1999.