# An Approach for Instance Based Schema Matching with Google Similarity and Regular Expression

Osama Mehdi, Hamidah Ibrahim, and Lilly Affendey
Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia

**Abstract**: *Instance based schema matching is the process of comparing instances from different heterogeneous data sources in determining the correspondences of schema attributes. It is a substitutional choice when schema information is not available or might be available but worthless to be used for matching purpose. Different strategies have been used by various instance based schema matching approaches for discovering correspondences between schema attributes. These strategies are neural network, machine learning, information theoretic discrepancy and rule based. Most of these approaches treated instances including instances with numeric values as strings which prevents discovering common patterns or performing statistical computation between the numeric instances. As a consequence, this causes unidentified matches especially for numeric instances. In this paper, we propose an approach that addresses the above limitation of the previous approaches. Since we only fully exploit the instances of the schemas for this task, we rely on strategies that combine the strength of Google as a web semantic and regular expression as pattern recognition. The results show that our approach is able to find* 1-1 *schema matches with high accuracy in the range of* 93%-99% *in terms of Precision* (*P*), *Recall* (*R*), *and F-measure* (*F*). *Furthermore, the results showed that our proposed approach outperformed the previous approaches although only a sample of instances is used instead of considering the whole instances during the process of instance based schema matching as used in the previous works.*

**Keywords**: *Schema matching, instance based schema matching, Google similarity, regular expression*.

*Received April 24, 2014; accepted August 31, 2015*

## 1. Introduction

One of the vital tasks in database integration is schema matching. At schema match core, discovering the correlation between database schemas is a serious challenge because of the syntactic and semantic heterogeneity [1, 4]. Matching two schemas S and T requires deciding if two attributes s of S and t of T represent the same real-world concept. While humans may be able to easily discover if two attributes match or non-match, however it is difficult for machines to discover it, especially when these two attributes have semantic heterogeneity. For example, s and t can represent different concepts but have the same name. The opposite is also possible; s and t can represent the same concept but have different names. During the process of schema matching, schema information which includes element name (schema name, attribute name), description, data type, constraint, and schema structure are normally used in an attempt to achieve correct matching between schemas. However, in some real world cases it may not be possible to use the information of schema structure. There are cases where information about the schema structure is not available such as in, fraud detection, crime investigation, counter-terrorism and homeland security De Carvalho *et al*. [8]. In such scenarios, instances are the only option available that can be used for schema matching.

Detecting instance correspondences could be a substitutional choice for schema matching, especially if the schema information is not available or might be available but worthless to be used for matching purpose. For example, it is common for a database designer to use abbreviations to represent schema attributes [23, 26, 28]. For instance, the attribute name CN could be an abbreviation of Customer Name or Company Name while SSN is an abbreviation of Social Security Number. For such case, only the instances can be used for determining the correspondences of attributes. Hence, the instances can give an accurate characterization of the actual contents of schema attributes [24, 26, 31].

By analysing the instance based schema matching approaches, we observed that neural network, machine learning, theoretic information discrepancy and rule based have been utilized. The goal of these approaches is to discover correspondences between schema attributes in which instances including instances with numeric values are treated as strings De Carvalho *et al*. [8]. This prevents discovering common patterns or performing statistical computation between the numeric instances. As a consequence, this causes unidentified matches especially for numeric instances and further reduces the quality of match results De Carvalho *et al*. [8].

In this paper, we propose an approach for instance based schema matching that aims at finding the

correspondences between schema attributes of two semantically and syntactically related data. Since we only explore the instances, we rely on strategies that combine the strength of Google as a web semantic and regular expression as pattern recognition to find the correspondences of schema attributes. As pointed out by [18, 22], there are different types of matching algorithms being applied in this area. However, this problem is still a research hotspot in order to further improve the accuracy of schema matching. Thus, our proposed approach is a step forward towards solving this problem.

The rest of this paper is organized as follows. Section 2 discusses the related works. Section 3 presents the proposed approach of instance based schema matching. In section 4, the evaluation metrics and the results are presented and discussed. Finally, section 5 draws the conclusions and points out the future work directions.

## 2. Related Work

This section presents a review of the previous approaches that have been proposed for the instance based schema matching to find correspondences between attributes of schemas. Neural network, machine learning, information theoretic discrepancy and rule based are approaches used for instance based schema matching.

### 2.1. Neural Network

The work in [18] classified attributes according to their field specifications and data values, and then trained a neural network for the recognition of similar attributes. This work assumes the similarity of structure and data values of attributes in different databases that represent the same real-world concept. Specific parsers of Database Management System (DBMS) are used to extract schema information or data contents from databases. Then, domain analysis compares the complete contents of each pair of attributes. The analysis starts by extracting a set of characteristics that describes the instances. The characteristics are divided into two types: character and numeric. A classifier is then used to discriminate attributes in a single database. The classifier output or cluster center then trains a neural network in recognizing categories and then determines similar attributes between databases. However, the back propagation algorithm requires that input characteristic has the same discriminators used in the training. They may not be able to obtain the same list of discriminators from two different databases. The average similarity of their work is 93%.

SEMantic INTegrator (SEMINT) [18] used neural network to study and identify correspondences among attributes in heterogeneous databases. It uses schema and instance information to automatically generate the matching rules of attribute. SEMINT also exploits multiple criteria that are constraint-based matching and content-based matching to find matching. SEMINT uses statistic computation for numerical field, while for character field, whose values are not computable as ASCII code numbers, SEMINT computes statistics on number of bytes actually used to store data. However, this method needs to be improved in terms of performance, especially in case of naming-based approach, as it achieved only 80% for precision.

Schema Matching method based on Data Distribution (SMDD) [20] introduces a schema matching method based on neural network, by analyzing the characteristics of data distribution. The SMDD algorithm believes that two attributes from heterogeneous data sources are similar if their data instance distributions are similar. SMDD has five steps as follows:

- *Step* 1. Involves parsing the instances to extract some instances for every attribute from the data source *S*1, analysing their characteristics of data distribution, and generating vectors from those characteristics.
- *Step* 2. Classifies the input vectors generated in step 1 into *M* categories using clustering algorithm.
- *Step* 3. Uses the output of step 2 as inputs for neural network in order to train the data.
- *Step* 4. Similarity determination is done in it.
- *Step* 5. The highest similar candidate mappings are selected.

The F-measure achieved by SMDD is between 0.25 and 0.65, while in the best case the F-measure can exceed 0.65.

A novel Content-Based Schema Matching Algorithm (CBSMA) [30] adopts neural network technique to perform instance-based schema matching task which fully explores the use of data content. This work introduces an innovative schema matching algorithm based on instances, which has two primary steps. The first step is the analysis of data pattern, which is done by training a set of neural networks. It starts with feature extraction, clustering to get training data and classifying data with Back Propagation Neural Network. Eleven features to describe the data pattern are proposed, the complete list of the features is given in [30]. The next step is applying some judgment rules to filter the candidate pairs and get the correct matching result. CBSMA achieved 96% and 90% of precision and recall, respectively.

### 2.2. Machine Learning

Learning Source Descriptions (LSD) [18] employs machine learning techniques that aims to semi automatically locates attributes matching. LSD requires initial examples of semantic mappings from the user, and employs these examples to train each

machine learning technique, whose solutions are combined and presented to the user. LSD first asks the user to provide the semantic mapping for a set of data sources, and then uses the mapping together with the sources to train a set of learners. Each leaner exploits a different type of information either in source schema or in their data. However, LSD achieved accuracy in the range of 71% - 92%.

Autoplex [2] is an instance based schema matching system. Autoplex used Naive Bayesian based learner to exploit characteristics of instances for identifying attributes match from a relational source schema to global schema. Naive Bayesian used example of instances to acquire probabilistic knowledge on the examples. The probability of instances occurs in an attribute mapped to another attribute, is estimated by the proportion of the occurrences of the instance among instances from attributes that are mapped to. For each attribute of the source schema, both match and mismatch probability with respect to every global attribute are determined. These probabilities are normalized to sum to 1 and the match probability is returned as the similarity between the source and global attributes. However, evaluation showed that Autoplex achieved only 0.81 for both soundness and completeness.

A new approach of instance based schema matching is proposed by [11], based on the hypothesis that corresponding attributes are relatively equally important. The main components of their three-part framework: attribute ranking, attribute classification and matching phase. In contrast to traditional approaches, which consider all attributes with the same importance, they employ machine learning methods in prioritizing all schema attributes according to rank and class. When matching, they have constructed an optimal objective function to determine all equivalent attributes. However, the approach is suitable only for numeric instances, as the result of precision (P) dropped to 66% when string instances are considered.

## 2.3. Information Theoretic Discrepancy

The approach by [21] handles the problem of schema matching when the interpretations of schema information are incorrect or ambiguous by evaluating the instances in schemas, playing as equivalent role as schema information. The approach has two-steps of domain-independent schema matching technique. In this technique shared information between pair-wise attributes employing the concept of mutual information is measured. Then, a graph is constructed to represent the weighted links for each input schema. Here, schema matching changes to a weighted graph matching problem and a graduated assignment algorithm is used to determine the correspondences of vertices between graphs. The proposed schema matching approach achieved 70% precision on average.

Two approaches proposed by [14, 15] are similar to the approach proposed by [21]. The approach in [15] involves computing the mutual information between pairs of columns in each schema, and then applying this statistical characterization of pairs of columns in one schema to propose matching pairs of columns in the other schema. The proposed approach has a two-step technique that can be used even when opaque column names and data values are present. First, it measures the pair-wise attribute correlations in the tables to be matched and then constructs a dependency graph utilizing mutual information as shown in Equation 1 as a measure of the dependency between attributes.

In the second step, it locates matching node pairs in the dependency graphs by running a graph matching algorithm.

$$I(X, Y) = \sum_i \sum_j \rho(X_i, Y_j) \log = \rho(X_i, Y_j) / \rho(X_i)\ \rho(Y_j) \quad (1)$$

- $I(X,Y)$ is the mutual information, or the strength of the relation of the attributes X and Y.
- $X_i$ and $Y_j$ are discrete values of $X$ and $Y$, respectively.
- $(X_i)$ and $(Y_j)$ are marginal probability when $X$ has the value of $X_i$ and $Y$ has the value of $Y_j$.
- $\rho(X_i, Y_j)$ is the joint probability.

The work in [15] is almost similar to their previous work in [13] except that it handles the remaining challenge from previous work which is the computational complexity of the graph-matching problem in the second step. As in the previous work of [14], this approach also has two steps. The first step measures the dependencies between attributes within tables using mutual information measure and constructs a dependency graph for each table to capture the dependencies among attributes. The second step involves finding matching node pairs across the dependency graphs utilizing graph matching algorithm. The P, R, and F achieved by [14, 15] is in the range of 45% - 93%.

## 2.4. Rule Based

Attribute identification is an algorithm proposed by [4] which fully explores data instances. It can also detect corresponding attributes to be integrated even in misleading schema information based on the assumption that entity identification can be performed successfully before matching. So, the main contribution of this study is to determine and offer an instance-based attribute identification method for the improvement of database integration in cases when schema information proves insufficient or unsuitable. However, their algorithm achieved 72% of matched attributes.

The work in [3] introduced algorithm that shows how the existence of duplicates in a data set would be

helpful and if exploited can automatically identify matching attributes. One of the rules that could be used "two attributes match if they have the same data values". The algorithm of [3] starts by discovering duplicates among data sets through comparison of instances in the duplicate records and then using them to perform schema matching between schemas and opaque column names. Soft-TFIDF is a measure used to determine the string similarity between the instances of the tuples. However, The P, R achieved by [3] are 75% and 87%, respectively.

From the previous works that have been reviewed in this section, it can be seen that different strategies have been applied in instance based schema matching. These works focused on one main goal which is to achieve high quality match results. The results reported by these works are in the range of 45%-96% in terms of the P, R, and F. However, most of these approaches treated the instances numeric values as strings. This prevents discovering common patterns or performing statistical computation between the numeric instances.

As a consequence, this causes unidentified matches especially for numeric instances and further reduces the quality of match results. Thus, much effort is still required to further improve the accuracy of instance based schema matching.

## 3. The Proposed Approach

In this section we present an instance based schema matching approach that determines correspondences between schema attributes. In addition, the proposed approach rely on strategies that combine the strength of Google as a web semantic and regular expression as pattern recognition. The proposed approach consists of five main phases as illustrated in Figure 1. These phases are analysing instances, classifying schema attributes, extracting the optimal sample size, identifying instance similarity, and identifying the match which are further explained in the following subsections.

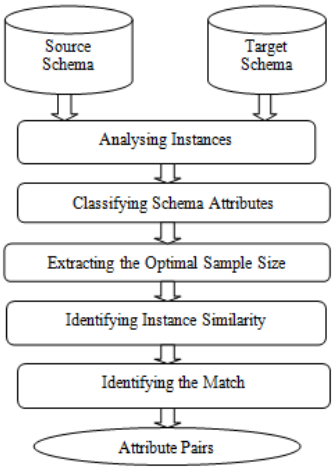


Figure 1. The phases of the proposed approach.

### 3.1. Analysing Instances

This phase aims to determine the data type of each attribute of both the target and source schemas. This is achieved by analysing the characters of an instance selected randomly from each attribute of the schemas. We classify the data type of an attribute as alphabetic, numeric and mix. The alphabetic data type is for attributes whose instances consist of only alphabetic characters ([A...Z, a…z]). The numeric data type is for attributes whose instances consist of only digit characters ([0…9]), whereas the mix data type is for attributes whose instances consist of combination of alphabetic, digit and special characters (e.g.,[-, /,\, ., ]). This phase starts by randomly selecting an instance of an attribute and counts the number of characters for each data type, then checks whether the number is equal to the length of the instance or not. If the number of characters of a data type is equal to the length of the instance (without whitespace), then the data type of the instance is identified as alphabetic if all the characters are alphabetic or numeric if all the characters are numeric. Otherwise, if the number of characters of a data type is less than the length of the instance, then the data type of the instance is identified as mix. For example, the instance "New York" has seven alphabetic characters which is equal to the length of the instance (without whitespace), while the instance "255 Courtland" has three numeric characters and nine alphabetic characters which are not equal to the length of the instance which is twelve. Thus, "New York" and "255 Courtland" are classified as alphabetic and mix data type, respectively.

### 3.2. Classifying Schema Attributes

After we have determined the data type of each attribute as discussed in the previous phase, the next step is to classify the attributes that share the same data type in the same class. The main aim of this phase is to reduce the number of possible comparisons that needs to be performed during the matching process. The maximum number of classes created for each schema is based on the number of data types that have been determined from the previous phase. Table 1 shows an example which is used to clarify this phase of the proposed approach. The following instances "New York", "Doctorate", "255 Courtland", "818/762-1221", and "49" have been classified into three data types which are alphabetic, numeric, and mix. Hence, three classes are created based on the identified data types. The class of alphabetic data type (*C_alpha*) includes the attributes of the instances "New York" and "Doctorate", the class of mix data type (*C_mix*) includes the attributes of the instances "255 Courtland" and "818/762-1221", and the class of numeric data type (*C_num*) includes the attribute of the instance "49".

Table 1. Classifying Schema Attributes based on the Data Type.

| Class of Alphabetic Data Type | |
|---|---|
| **Attribute 1** | **Attribute 2** |
| New York | Doctorate |
| Class of Mix Data Type | |
| **Attribute 1** | **Attribute 2** |
| 255 Courtland | 818/762-221 |
| Class of Numeric Data Type | |
| **Attribute 1** | - |
| 49 | - |

## 3.3. Extracting the Optimal Sample Size

This phase aims at extracting a sample of instances for each attribute of the classes (C_alpha, C_num, and C_mix) based on the optimal sample size which is equal to 50% of the actual table size. The optimal sample size has been chosen through a set of experiments. This is due to the fact that relying on the entire instances of an attribute which might involve huge data size to determine the similarity between attributes will lead to low performance with respect to processing time. The experiments of the optimal sample size have been designed in such a way that each experiment will use different size of samples starting from 5% of the actual table size. The size of samples is increased either 5% or 10% in the subsequent experiments. The experiments are ended when the P, R, and F are at least 96% which is at least equal to the best results reported in the previous works [30].

## 3.4. Identifying Instance Similarity

The aim of this phase is to compare the attributes in the same class that belong to different schemas, whether they are representing the same entity or not. To find correspondences between attributes in each class, two strategies are adopted. The first strategy utilizes regular expression for syntactic similarity and the second strategy utilizes Google similarity for semantic similarity.

### 3.4.1. Regular Expression

Regular expression (known as regexes) is a way to describe text through pattern (format) matching and provide an easy way to identify text. Regular expression is a language used for parsing and manipulating text [12, 17]. Furthermore, regular expression is a string containing a combination of normal characters and special metacharacters or metasequences (*, +, ?). Table 2 shows the most common metacharacters and metasequences in regular expression [10] that are used in our work.

Table 2. The Common Metacharacters in Regular Expression.

| Metacharacter | Name | Matches |
|---|---|---|
| d\ | Digit | Matches a digit |
| s\ | Whitespace | Matches whitespace |
| [a-z, A-Z] | A range of letters | Matches any letter in the specified range |
| . | Dot | Matches any one character |
| […] | Character class | Matches any one character listed |
| [^…] | Negated character class | Matches any one character not listed |
| ? | Question | One allowed, but it is optional |
| * | Star | Any number allowed, but all are optional |
| + | Plus | At least one required; additional are optional |
| \| | Alternation | Matches either expression it separates |
| ^ | Caret | Matches the position at the start of the line |
| $ | Dollar | Matches the position at the end of the line |
| {X,Y} | Specified range | X required, max allowed |

Regular expression provides several benefits, which are [9, 27]:

- Relatively inexpensive and does not require training or learning as in learning-based or neural network techniques.
- Regular expression can provide a quick and concise method to capture valuable user knowledge about the domain.

In our approach, regular expression is derived only for those attributes with numeric or mix data types. This is explained below.

#### 3.4.1.1. Regular Expression for Numeric Data Type

This sub-section explains the process of creating regular expression for the attributes with *numeric* data type. The attributes with *numeric* data type are attributes whose instances consist of only digit characters ([0…9]). In creating a regular expression for an attribute, the minimum and maximum values of the attribute are required. Thus, three variables have been identified, namely: *nomin*, *nomax* and *uppervalue*. Initially, *nomin* and *nomax* are assigned the minimum and maximum values of the attribute, respectively. However in the following iterations, the value of *nomin* is changed to the last *uppervalue* + 1. The *uppervalue* is a value which is greater than the value of *nomin* and less than the value of *nomax*; and is derived based on the following conditions:

a) When the *nomin's* length of digits is less than the *nomax's* length of digits, the *uppervalue* is the maximum value based on the *nomin's* length of digits and not greater than the value of *nomax*. For instance, if the *nomin's* length of digits is three (e.g., 345) then the *uppervalue* is 999. If the *uppervalue* is greater than the value of *nomax*, then

the first digit of the *uppervalue* is changed to the first digit of *nomin* (399 for the above example). This is then checked against the value of *nomax*. If the new *uppervalue* is still greater than the value of *nomax* then the second digit of the *uppervalue* is changed to the second digit of *nomin* (349 for the above example). This process is repeated in which the next digit of the *uppervalue* is changed to the next digit of *nomin* until the condition stated in the definition of *uppervalue* is satisfied. However, if all the digits of *uppervalue* have been changed, i.e., the value of *uppervalue* is now equal to the value of *nomin*, then the value of *nomax* is assigned to *uppervalue*. This is to reduce the number of iterations needed in identifying the *uppervalue*.

b) When the *nomin's* length of digits is equal to the *nomax's* length of digits and the *nomin* has at least one zero digit on the right, the *uppervalue* is derived using the formula shown in Equation 2. The Equation 2 derives the closest *uppervalue* to the *nomax*.

$$uppervalue = (nomax - (nomax\ MOD\ Sumz * 10) - 1) \quad (2)$$

Where *Sumz* returns the number of zero digits on the right of the *nomin*. If the Equation 2 returns an *uppervalue* which does not satisfy the condition that we have stated earlier, then the step as mentioned in (a) above is applied. For instance, when the *nomin's* length of digits is equal to the *nomax's* length of digits (e.g. 120 and 123, respectively), the *uppervalue* is 119 based on the Equation 2. The result of *uppervalue* does not satisfied the definition of *uppervalue* which is greater than the value of *nomin* and less than the value of *nomax*. Then, the steps as mentioned in condition (a) above are applied to derive the value of *uppervalue*. Table 3 illustrates an example of the proposed idea for numerical data type.

Table 3. The Mechanism of the RegEx for Numerical Domain.

| Iteration | Nomin | Upper value | RegEx | Accumulated RegEx |
|---|---|---|---|---|
| 1 | 7 | 9 | [7-9] | [7-9] |
| 2 | 10 | 99 | [1-9][0-9] | [7-9]\|[1-9][0-9] |
| 3 | 100 | 119 | 1[0-1][0-9] | [7-9]\|[1-9][0-9]\|1[0-1][0-9] |
| 4 | 120 | 123 | 12[0-3] | [7-9]\|[1-9][0-9]\|1[0-1][0-9]\|12[0-3] |

#### 3.4.1.2. Regular Expression for Mix Data Type

The attributes with *mix* data type consist of instances that includes alphabetic, numeric and special characters. The general idea is to divide an instance into a set of sub-tokens. Each sub-token is a sequential set of characters of a particular data type. Then, a regular expression is built for each sub-token of the instance. Finally, the regular expressions of each sub-token are combined as the regular expression of the instance. For example, the following instance "255 Courtland" can be divided into two sub-tokens which are "255" and "Courtland". The first sub-token "255" is considered as a sub-token of the *numeric* data type since it consists of a sequential set of numeric characters. While, the second sub-token "Courtland" belongs to the *alphabetic* data type as it consists of a sequential set of *alphabetic* characters. Finally, we combine the regular expressions of each sub-token that are "\\d+" for the sub-token with numeric characters and "([a-zA-Z]+)" for the sub-token with the alphabetic characters as the final regular expression of the instance "255 Courtland". Table 4 illustrates an example of the proposed idea for mix data type.

Table 4. An example of the mix data type

| Instance | Regular Expression |
|---|---|
| 255 Courtland | d\+\s[a-z, A-Z]+ |
| 589/265/954 | d\+/d\+/d\+ |

### 3.4.2. Google Similarity

The Google similarity uses the World Wide Web as a database and Google as a search engine. Google's similarity of words and phrases from the World Wide Web uses Google page counts, as shown in Equation 3.

$$GSD(x, y) = max(\log f(x), \log f(y)) - \log f(x, y) \,/\, \log M - min(\log f(x), \log f(y)) \quad (3)$$

Where $f(x)$ is the number of Google hits for the search term $x$, $f(y)$ is the number of Google hits for the search term $y$, $f(x, y)$ is the number of Google hits for both terms $x$ and $y$ together, and $M$ is the number of web pages indexed by Google. The World Wide Web is the largest database on earth and the context information entered by millions of independent users averages out to provide automatic semantics of useful quality [6, 7]. The Google similarity calculates the semantic similarity score for the attributes with alphabetic data type that comprises instances consisting of only alphabetic characters ([A...Z, a…z]). For instance, if we want to search for a given term in the Google web pages, e.g. "Msc", we will get a number of hits that is 127,000,000. This number refers to the number of pages where this term is found. For another term, "Phd", the number of hits for this term is 50,600,000. Furthermore, if we search for those pages where both terms " Msc" and "Phd" are found, that gives us 36,100,000 hits. Consequently, we can use these numbers of hits for the terms "Msc", "Phd" and both terms together in addition to the number of pages indexed by Google, which is around 3,000,000,000 in the Equation 3. The equation produces the similarity degree between the two terms "Msc" and "Phd" as follows: *GSD* (Msc, Phd) = 0.31.

## 3.5. Identifying the Match

After we have analyzed the instances, classified the attributes, extracted the optimal sample size, and performed the tasks of syntactic and semantic matching, the last phase of our proposed approach

attempts to find the correct matching between the attributes that shared the same data type. For classes of numeric and mix, this phase specifies a match by matching the regular expression of the instances of the source schema derived from the previous phase against sample of instances of the target schema. If a match occurs, then the instances of the regular expression of the source schema is said to correspond to the instances of the target schema. Finally, for the class of alphabetic we compare the similarity scores of the instances that have been derived from the previous phase (utilizing Google similarity) with a predefined threshold value. In our work the threshold value is set to 60 as used by previous work Khan *et al*. [16]. Similarity scores for instances above or equal to the threshold are related; otherwise they are not related.

## 4. Results and Discussion

### 4.1. Data Set

For the purpose of evaluating our proposed approach two real data sets have been used in the experiment study, namely: Restaurant [25] and Census [29], both of which are available online. In our experiments we created two sub-tables by randomly selecting the attributes from the original table of both data sets. The number of attributes of each sub-table is equal to the number of attributes of the original table. However, these attributes might occur in different sequence and the same attributes might be selected more than once. These sub-tables were populated with instances taken randomly from the original table of the data sets. The number of instances of both sub-tables is different to represent real world cases. We pretended that these sub-tables were two different tables that needed to have their schemas matched as applied in [5, 14, 15].

Restaurant is the first real data set to which our approach is applied. The data set comprises of lists of restaurants in two popular websites that are Zagat and Foodor. Restaurant data set has 864 records and five attributes, namely: Name, Address, City, Phone Number, and Type of Food. The attribute Name is the name of the restaurant that contains instances comprising of only letters. The attribute Address refers to the address of the restaurant, this field contains mix of characters (numeric and alphabetic). The attribute City refers to the city of the restaurant. The attribute Type of Food refers to the type of food provided by the restaurant. Both the attributes Type of Food and City contain instances which comprise of only letters. Finally, the attribute Phone Number contains numbers and some special characters such as '/' and '-'. Restaurant data set has been used as it is one of the real databases which is available and appropriate to evaluate our proposed approach.

For our second data set, we used Census data set which contains 358171 instances, 32561 records and 11 attributes. Four attributes consist of only numeric characters, which are *age*, *fnlwgt*, *Education-num* and *capital-gain*, while the other attributes contain only letters, which are *workclass*, *education*, *relationship*, *race*, *sex*, *marital status*, and *native-country*. Census data set have been used by several proposed approaches [14, 15].

For comparison purpose, we compared our proposed approach to [14, 15] in terms of P, R, and F. We did not compare our proposed approach to the approaches that are reported in the related work section, as these approaches used different data sets that are not accessible through the internet [3, 11, 15, 19, 21, 22, 30]. Besides, some of these approaches required specific rules [3, 21] and user intervention [17, 18, 27] to perform the matching process.

### 4.2. Measurements

The evaluation metrics considered in this work are P, R and F shown in Equations 4, 5 and 6, respectively. It is based on the notion of true positive, false positive, true negative, and false negative.

- True Positive (TP):The number of matches detected when it is really matches.
- False Positive(FP):The number of matches detected when it is really non-match.
- True Negative (TN): The number of non-matches detected when it is really non-match.
- False Negative (FN): The number of non-matches detected when it is really matches.

$$Precision = |TP| / |TP| + |FP| \quad (4)$$

$$Recall = |TP| / |TP| + |TN| \quad (5)$$

$$F\text{-}measure = 2 * Precision * Recall / Precision + Recall \quad (6)$$

For each table, we kept the number of attributes to 11 and 5 for Census and Restaurant data sets, respectively. We repeated each experiment 5 times, measured the *P*, *R* and *F* and averaged these results.

### 4.3. Result

Figure 2 presents the results of accuracy in terms of *P*, *R* and *F* for the proposed approach of instance based schema matching. From the results presented in Figure 2, the following can be concluded: 1) we achieved 96% for *P*, 93% for *R* and 95% for *F* for the Restaurant data set, while with Census data set we achieved 99% for *P*, 96% for *R*, and 97% for *F*. The size of samples used is 50% of the actual table size which has been identified through experiments; and 2) our proposed approach produced high accuracy in spite of the approach considered a sample of instances instead of considering the whole instances during the process of instance based schema matching.

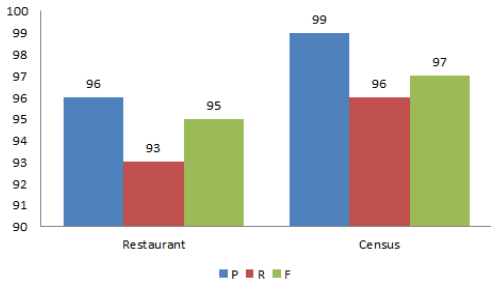


Figure 2. Matching results of census and restaurant data sets.

Figure 3 shows the matching results using census data set of our proposed approach compared to the approaches proposed by [14, 15] in terms of P, R and F. From these results, we can conclude that our proposed approach achieved better results although only a sample of instances is used instead of considering the whole instances during the process of instance based schema matching as used in the previous works [14, 15].

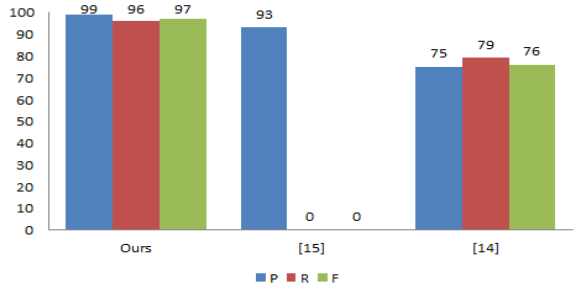


Figure 3. Matching results of the census data set.

## 5. Conclusions

In this paper, we proposed an instance based schema matching approach to identify 1-1 schema matching. Our proposed approach rely on strategies that combine the strengths of Google similarity as a web semantic and regular expression as pattern recognition. Our experimental results show that our proposed approach is able to identify 1-1 matches with high accuracy in terms of *P*, *R* and *F* although only a sample of instances is used instead of considering the whole instances during the process of instance based schema matching. In the near future, we plan to extend our framework to handle complex schema matching (n-m), since identifying complex matches is a more challenging problem.

## References

[1] Benslimane S., Malki M., and Bouchiha D., "Deriving Conceptual Schema from Domain Ontology: A Web Application Reverse Engineering Approach," *The International Arab Journal of Information Technology*, vol. 7, no. 2, pp. 167-176, 2010.

[2] Berlin J. and Motro A., *Cooperative Information Systems*, Springer Link, 2001.

[3] Bilke A. and Naumann F, "Schema Matching using Duplicates," *in Proceeding of the 21st International Conference on Data Engineering*, Washington, pp. 69-80, 2005.

[4] Christen P, *Data Matching*: *Concepts and Techniques for Record Linkage, Entity Resolution and Duplicate Detection*, Springer Link, 2012.

[5] Chua C., Chiang R., and Lim E., "Instance-based Attribute Identification in Database Integration," *The International Journal on Very Large Data Bases*, vol. 12, no. 3, pp. 228-243, 2003.

[6] Cilibrasi R., and Vitanyi P., "The Google Similarity Distance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 370-383, 2007.

[7] Cilibrasi R. and Vitanyi P., "Automatic Meaning Discovery using Google," Technical Report, 2004.

[8] De Carvalho M., Laender A., Gonçalves M., and Da-Silva A., "An Evolutionary Approach to Complex Schema Matching," *Information Systems*, vol. 38, no. 3, pp. 302-316, 2013.

[9] Doan A. and Halevy A., "Semantic Integration Research in the Database Community: A Brief Survey," *AI Magazine*, vol. 26, no. 1, pp. 83-94, 2005.

[10] Doan A., Domingos P., and Halevy A., "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach" *in Proceedings of ACM SIGMOD International Conference on Management of Data*, New York, pp. 509-520, 2001.

[11] Feng J., Hong X., and Qu Y., "An Instance-Based Schema Matching Method with Attributes Ranking and Classification," *in Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery*, New Jersey, pp. 522-526, 2009.

[12] Friedl J., *Mastering Regular Expressions*, *O'Reilly Media*, 2006.

[13] Goyvaerts J. and Levithan S, *Regular Expressions Cookbook*, *O'reilly*, 2012.

[14] Kang J. and Naughton J., "On Schema Matching with Opaque Column Names and Data Values," *in Proceeding of the ACM SIGMOD International Conference on Management of Data*, New York, pp. 205-216, 2003.

[15] Kang J. and Naughton J., "Schema Matching using Interattribute Dependencies," *Knowledge and Data Engineering IEEE Transactions*, vol. 20, no. 10, pp. 1393-1407, 2008.

[16] Khan L., Partyka J., Parveen P., Thuraisingham B., and Shekhar S., "Enhanced Geographically-Typed Semantic Schema Matching," *Journal of Web Semantics*, vol. 9, no. 1, pp. 52-70, 2011.

[17] Kleene S., *Representation of Events in Nerve Nets and Finite Automata*, Princeton University Press, 1951.
[18] Li W. and Clifton C., "SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases using Neural Networks," *Data and Knowledge Engineering*, vol. 33, no. 1, pp. 49-84, 2000.
[19] Li W. and Clifton C., "Semantic Integration in Heterogeneous Databases using Neural Networks," *in Proceeding of the 20th International Conference on Very Large Data Bases*, San Francisco, pp. 1-12, 1994.
[20] Li Y., Liu D., and Zhang W., "Schema Matching using Neural Network," *in Proceeding of the IEEE/WIC/ACM International Conference on Web Intelligence*, Washington, pp. 743-746, 2005.
[21] Liang Y., "An Instance-Based Approach for Domain-Independent Schema Matching," *in Proceeding of the 46th Annual Southeast Regional Conference on XX*, New York, pp. 268-271, 2008.
[22] Liu G., Huang S., and Cheng Y., *Frontiers in Computer Education*, Springer, 2012.
[23] Mehdi O., Ibrahim H., and Affendey L., "Instance Based Matching using Regular Expression," *Procedia Computer Science*, vol. 10, pp. 688-695, 2012.
[24] Rahm E. and Bernstein P., "A Survey of Approaches to Automatic Schema Matching," *The International Journal on Very Large Data Bases*, vol. 10, no. 4, pp. 334-350, 2001.
[25] Restaurant Reviews Dataset, http://www .cs.cmu.edu/~ mehrbod/RR/., Last Visited 2014.
[26] Riaz M. and Munir S., *An Instance Based Approach to Find the Types of Correspondence Between the Attributes of Heterogeneous Datasets*, Isseratration Academic Book Publishers, 2012.
[27] Stubblebine T., *Regular Expression Pocket Reference*: *Regular Expressions for Perl*, *Ruby*, *PHP*, *Python*, *C*, *Java and. NET*, Amazon, 2007.
[28] Tejada S., Knoblock C., and Minton S., "Learning Object Identification Rules for Information Integration," *Information Systems*, vol. 26, no. 8, pp. 607-633, 2001.
[29] UCI Machine Learning Repository, http://archive. ics.uci.edu/ml/ datasets.html, Last Visited 2014.
[30] Yang Y., Chen M., and Gao B., "An Effective Content-Based Schema Matching Algorithm," *in Proceeding of the International Seminar on Future Information Technology and Management Engineering*, Washington, pp. 7-11, 2008.
[31] Zaib K., "Instance-Based Ontology Matching and the Evaluation of Matching Systems," PhD Dissertation, Dusseldorf University.

**Osama Mehdi** received his Bachelor of Computer Science from the University of Babylon, Iraq in 2009 and M.Sc. by research degree in computer science and information technology from University Putra Malaysia, Malaysia in 2014. Currently, he is working as a lecturer at Al Mustaqbal College University. His research interests include Data Integration, Information Retrieval, Semantic Web, Pattern Recognition and Large-Scale Data Analysis (Big Data).

**Hamidah Ibrahim** is currently a professor at the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. She obtained her PhD in computer science from the University of Wales Cardiff, UK in 1998. Her current research interests include databases (distributed, parallel, mobile, bio-medical, XML) focusing on issues related to integrity constraints checking, cache strategies, integration, access control, transaction processing, and query processing and optimization; data management in grid and knowledge-based systems. (e-mail: hamidah.ibrahim@upm.edu.my).

**Lilly Affendey** received her Bachelor of Computer Science from the University of Agriculture, Malaysia in 1991 and MSc in Computing from the University of Bradford, UK in 1994. In 2007 she received her PhD in Database Systems from University Putra Malaysia. Her research interests are in Multimedia Database, Content-based Video Retrieval and Big Data Analytics. She is currently an Associate Professor in University Putra Malaysia.