Research Journal of Applied Sciences, Engineering and Technology 7(3): 454-461, 2014
DOI:10.19026/rjaset.7.275
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: May 07, 2013 Accepted: June 11, 2013 Published: January 20, 2014

Research Article

Ontology Matching Algorithm by Using Agent Technology

Abdulsalam Alarabeyyat
Faculty of Information Technology, Al Balqa Applied University, Salt, Jordan

Abstract: In this study we Introduce a three dimensional algorithm for ontology matching problem by comparing ontologies concepts from three dimensions (i) matching concepts based on name similarity (ii) matching concepts based on content similarity (iii) matching concepts based on relationship similarity. By comparing concepts from three dimensions we can trust our result since; the same concepts may be named by different labels or they may have the same names but differ in their attribute or their relations. In these cases if we rely on one dimension in the matching process; we will not discover the similarities. On the contrary, when the matching process relies on three dimensions, if one dimension can't discover the similarity the other will do.

Keywords: Ontlogy, ontlogy matching, ontlogy matching algorithm, software agent

INTRODUCTION

The main subject of this study is to study ontology matching, the task of identifying the correspondences between elements of two ontologies, where solving such match problems has a significant importance in different application domains.

In order to have a comprehensive view of the work presented in this study, it is necessary to take a view on terms such as agents, ontology, ontology matching and multi-agents system. Agents are computational systems that are capable of autonomous, reactive and proactive behavior endowed with the ability to interact with other agents (Fasli, 2007).

Software agents differ from other traditional software applications, where traditional software applications need to be told explicitly what it is that they need to accomplish and the exact steps that they have to perform, agents need to be told what the goal is but not how to achieve it. Then being smart, they will actively find ways to satisfy this goal, acting with the minimum intervention from the user. Agents will determine what needs to be done to achieve their goal, but also react to any changes in the environment as they occur, which may affect their plans and goals, accomplishments, then according to these changes they will modify their course of action. An agent is usually not an isolated entity; it is located within an environment and continuously interacts with it as well as with other entities, including agents and humans. A multi-agent system consists of number of agents that communicate and interact with each other to solve a complex problem or to achieve their user's goals.

For agents to be able to communicate with and understand each other they need to use results of all areas of artificial intelligence such as learning, negotiation or knowledge management. This study focuses on one of them-a use of knowledge management by agent systems. During negotiations the agents use the knowledge about the problem domain in order to fulfill their goals.

Most ontology matching techniques belong to either rule based or learning-based ones. Cupid Madhavan *et al*. (2001) proposed a mapping algorithm between schema elements based on their names, constraints, data types and schema structure. Cupid has a bias toward leaf structure where much of the schema content resides.

Madhavan *et al*. (2005) describes how a corpus of schemas and mappings can be used to match schemas. Such a corpus typically contains multiple schemas that model similar schemas concepts and their properties. First they increase the evidence about each concept being matched by including evidence from similar concepts in the corpus. Then they learn statistics about concepts and their relationships to infer constraints.

Williams (2004) proposed an algorithm for multiagent knowledge sharing and learning in a peer-to-peer setting. It enables multiagent systems to assist groups of people in locating, translating and sharing knowledge. After locating similar concepts, agents can continue to translate concepts and then are able to share meanings and learning and translating similar semantic concepts between them.

PROBLEM STATEMENT

Depending on our knowledge, ontology defines a common vocabulary for those who need to share information in a domain. In other words, ontology represents a common ground for those wishing to enter

in meaningful interactions. This is one of the primary reasons why we need to develop ontologies.

While ontologies serve as a basis for solving this problem, the heterogeneity among different ontologies considered to be a serious problem. This is due to the fact that the domains are developed by different parties that can design ontologies according to their own conceptual views of that domain.

Confusion and misunderstanding may arise on the following cases:

- The same names do not necessarily indicate the same semantics and different names may be used to represent the same real-world concept. If agents refer to the same concepts by employing different terms as an example, agent *i* refer to anyone who buys products as a *client* whereas agent *j* refers to such an entity as a *customer*. Thus, the same thing can be described in different ways, each representing different perspectives or ways of thinking.

Confusion may also arias when agents refer to different concepts using the same term. For agent *i* an employee is anyone in the payroll system, whereas for agent j an employee is anyone receiving benefits. If agents are not aware of this distinction, they will not be able to enter into a meaningful dialogue. In addition Element names may be encrypted or abbreviated so that they are only understandable by their creators. Also building Ontologies by different languages and different conceptual modeling can be used.

PROPOSED SOLUTION

Heterogeneity alleviating will be achieved by providing mechanisms in which intelligent agents that work in open environments could communicate in an efficient way, even in the cases when they use completely different ontologies. To handle this heterogeneity issue in ontologies, many approaches have been proposed. Generally, there are two different kinds of solutions (Huang *et al*., 2007):

- **Centralized solution:** In which a central ontology that is agreed-upon, global and unique and includes every concept that can satisfy the needs of different parties is built. However, a central ontology will never be large and compatible enough to include all concepts of interest to every individual ontology designer, so it will have to be modified and extended. Each new extension will be different and increase incompatibility.
- **Distributed solution:** This solution focuses on the ability for individual ontologies to match and reconcile with each other and possibly reuse each other. The ontology matching is initially carried out by a human. There are some drawbacks to this manual process, including that it is time-consuming and error-prone, as ontologies can contain thousands of concepts. Therefore, this solution focuses on developing tools that are either automatic or semi-automatic and can help people in matching ontologies.

Algorithm structure: The Three-Dimensional algorithm matches two ontologies *O*1 and *O*2, depending on three major steps:

- Matching concepts based on name similarity
- Matching concepts based on content similarity
- Matching concepts based on relationship similarity

We have defined ontologies as "a formal representation of a set of concepts within a domain, properties of each concept and the relationships between those concepts". We have suggested our algorithm depending on this definition, since we say that these three features together specify a concrete view for each concept:

- The name of concept
- The properties of concept
- The relationships of the concept

By comparing different concepts according to these three features we can trust our result. For example, some concepts that give the same meaning may differ in their names in this case we can match them by relying on the similarity on their properties and the relationship among them. On contrary, some matchers that rely on concept name similarity will fail in this case since the names are different. Also we have used learning techniques in our algorithm.

We have summarized our contribution on the following:

- Building a three dimensional algorithm for ontology matching.
- Our algorithm integrates an Artificial Neural Network (ANN) technique.
- Integrating syntactic and semantic matching in our matching algorithm.

Parameters used in the algorithm: Ontology matching is defined as "a function *f*, which, from a pair of ontologies to match *O* and *O*~, a set of parameters *p* and a set of resources *r*, returns an alignment A between these ontologies". In this section we will explain the two parameters that used by the algorithm and how these parameters will have a significant influence on the final result.

Similarity threshold: The similarity threshold parameter dictates a minimum similarity values

between two concepts in order to consider them as a matched concepts. We have used two similarity thresholds in our algorithm, the first one in content matching step; at this step we consider for any two compared attributes that having a name similarity greater than a threshold which we determine to be 0.60 they will matched from content dimension. Also we have another threshold, for the final results which are obtained by a weighted sum of *sN*, *sC* and *sR*. If this final results is greater than 0.30 then the compared concepts will be considered as a matching concepts.

We have determined these thresholds by trial-and-error. As an example, when we compare two attributes LCD Screen Size to Screen Size we will have a similarity value equals to 0.76 and so they will consider to be matched, since their similarity value is greater than the specified threshold.

Similarity weights: During the matching process different aspects, i.e., concept names, concept properties and concept relationships, contribute in different degrees to the matching result. So we need to assign weights to these aspects according to their importance, a more accurate matching result is favored.

We will use machine learning techniques, such that the weights can be learned from training examples instead of being ad-hoc defined by domain experts, the learning process will be achieved by using Artificial Neural Network (ANN) (Huang *et al*., 2007). On the following we will explain the neural network design and the learning process.

Neural network design: We build a two-layer 3×1 network, as shown in Fig. 1. On the following we will summarize the neural network characteristics.

Network inputs:

sN : Similarity in name
sC : Similarity in contents
sR : Similarity in relationship

Network outputs:

*O*1 : *sN* multiplying by weight (w1)
*O*2 : *sC* multiplying by weight (w2)
*O*3 : *sR* multiplying by weight (w3)

Target function: Each pair of compared concepts correspond to *cell* (*i*, *j*) in *O*1, *O*2, *O*3; Target function tries to find the maximum value of row (*i*) and column (*j*).

Training set: We will randomly take a set of concepts from source ontology (Products) and find the equivalent concepts by manual matching with target ontology (Electronic equipments).

Training process: The training process consists of the following main steps:

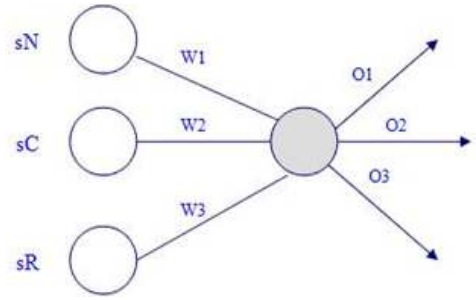

Fig. 1: Neural network structure

Step 1: Initialization: Set initial weights randomly:

$W1 = 0.27, W2 = 0.40, W3 = 0.33$

Step 2: Activation: Activate the network by applying inputs (SN, sC and sR) and the randomly initialized weights and then calculate the actual outputs.

Step 3: Error Calculation:

$$E(wi) = [((max\_row) - Oi) + ((max\_column) - Oi)]$$

Step 4: Weight training:

$$w1N = w1 + \alpha * \Sigma\ [((max\_row) - O1) + ((max\_column) - O1)] * sN$$

$$w2N = w2 + \alpha * \Sigma\ [((max\_row) - O2) + ((max\_column) - O2)] * sC$$

$$w3N = w3 + \alpha * \Sigma\ [((max\_row\ ) - O3) + ((max\_column) - O3)] * sR$$

We randomly set $w1 = 0.35$, $w2 = 0.32$, $w3 = 0.33$, learning rate α is set to 0.2 and then we randomly pick up a set of concepts from product ontology and find the corresponding equivalent concepts by a manual matching with electronic equipments ontology. Each of such manually matched pairs will be processed by the network and the similarity values in name, properties and relationships for these two concepts are calculated and used as a training example to the network. We then use the learned weights ($w1 = 0.27$, $w2 = 0.40$, $w3 = 0.33$) to match the remaining concepts.

Resources used in the proposed algorithm: In order to identify mappings between the concepts of ontology, called the source ontology with concepts of another one, called the target ontology, a lot of recent works use additional resource called background knowledge. The common objective for using an external resource is to detect desired matches which may fail in some cases. In our algorithm we will use *WordNet* as background knowledge.

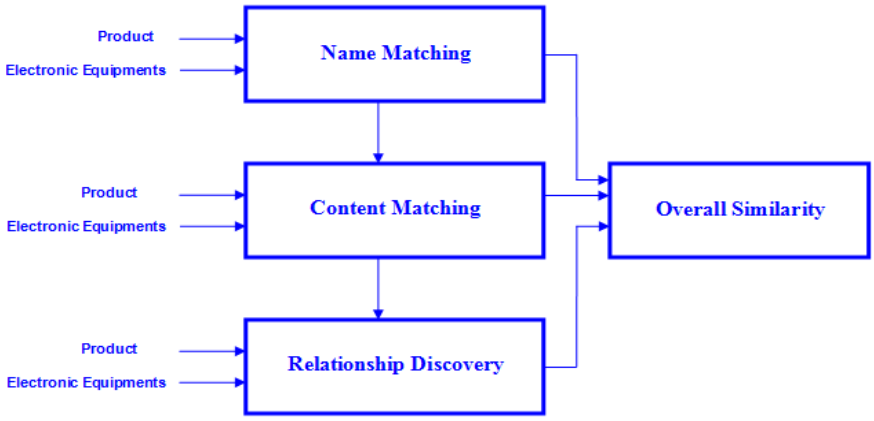

Fig. 2: Ontology matching process

Table 1: Relations in *Word Net*

Relations	Description	Example
Hypernym	Is a generalization of	Motor vehicle is a hypernym of car
Hyponym	Is a kind of	Car is a hyponym of motor vehicle
Meronym	Is a part of	Lock is a meronym of door
Holonym	Contains part	Door is a holonym of lock
Troponym	Is a way to	Fly is a troponym to travel
Antonym	Opposite of	Stay in place is an antonym or travel
Attribute	Attribute of	Fast is an attribute of speed
Entailment	Entails	Calling on phone entails dialing
Cause	Cause to	To hurt cause to suffer
Also see	Related verb	To lodge is related to reside
Similar to	Similarly to	Evil is similar to bad
Participle of	Is participle of	Stored is the participle of to store
Pertainym	Pertains to	Radial pertains to radius

Word Net Yatskevich and Giunchiglia (2004) is a lexical database which is available online and provides a large repository of English lexical items. Word Net contains synsets (or senses), structures containing sets of terms with synonymous meanings. Each synset has a gloss that defines the concept that it represents. For example, the words night, nighttime and dark constitute a single synset that has the following gloss: the time after sunset and before sunrise while it is dark outside. The relations of Word Net are presented on Table 1.

Matching process: The proposed algorithm will take as input two ontologies which intend to match and computes as output a set of matched elements in three major steps, firstly calculating similarities in names then the similarities in contents are computed, which include comparing for the data type and attributes of the concepts and finally discovering the relationships between compared concepts. For each step the process of calculating the similarities for all concepts (one from the first ontology and the other from the second one, considering all combinations) is done by building an $m1 \times m2$ matrix to record all the calculated values, where $m1$ is the number of concepts in the first ontology and $m2$ is the number of concepts in the second one and the value in *cell* $[i, j]$ store the similarity between the i^{th} concept in first ontology and the j^{th} concept in the second, then the overall similarities between all concepts from two ontologies is given by the summation for the results of multiplying each similarity matrix by the assigned weight to it. Figure 2 clarifies our matching process.

Direct ontology matching: In steps 1 and 2, we generate the similarities depending on direct matching, that is we depend only on the information contained on the ontologies we are interested to match.

The first step of our algorithm attempts match concepts between two ontologies by measuring similarities between their names. We will find the similarities between concepts names depending on edit distance function which calculate the minimum number of edit operations required to transform one string into the other. Most commonly, the edit operations allowed for this purpose are:

- Insert a character into a string
- Delete a character from a string
- Replace a character of a string by another character

for these operations, edit distance is sometimes known as *Levenshtein* distance. We have used edit distance Chirstopher *et al*. (2009) for the following features which show the importance of edit distance comparing with other available functions:

- Edit distance can find the distance between concepts of different length.
- Edit distance gives more realistic and flexible results comparing with some lexical matchers such as string equality matcher and substring matcher, which give 1 for the similar words and 0 for non similar without saying anything about the degree of similarity or dissimilarity.
- Edit distance can find similarities in any part of the word, on the contrary to some matchers that are able to find the similarities in prefix or in suffix of the word only.

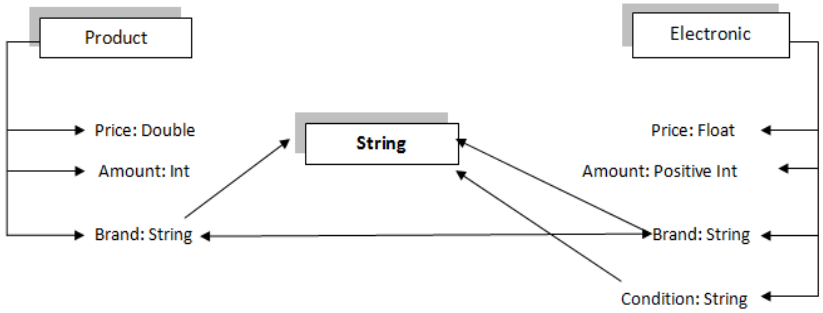

Fig. 3: Content matching

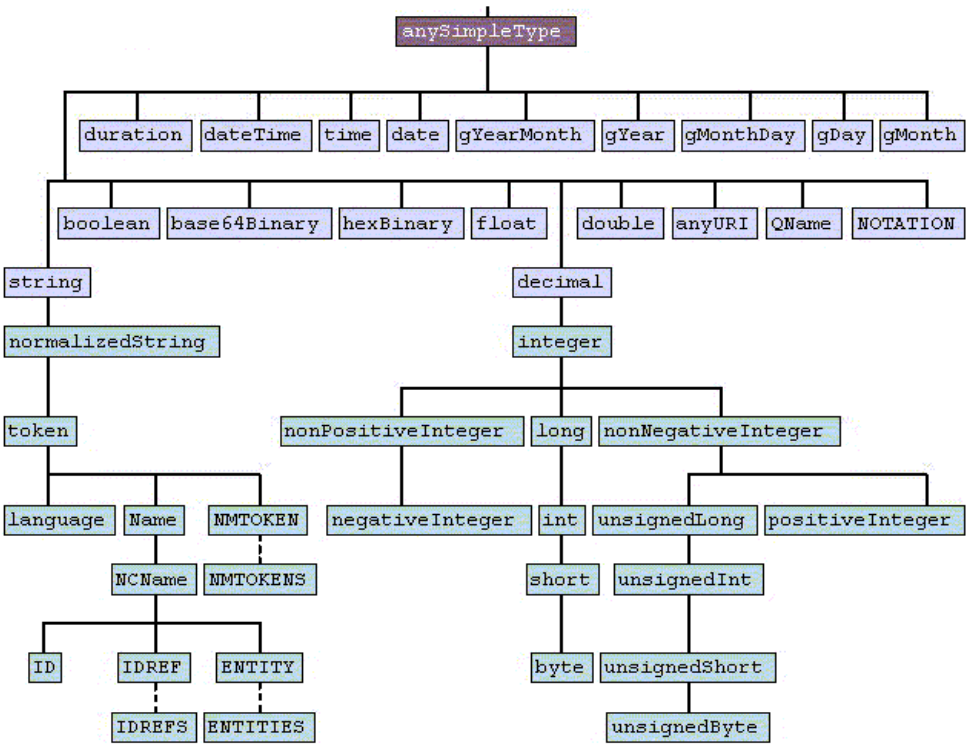

Fig. 4: XML schema built-in datatype (Biron *et al*., 2004)

Similarities in names are calculated according to the following function:

$$sN = 1 - (Edit\_Distance / (max\_length)) \quad (1)$$

where, sN is the name similarity and has a real value in the range of [0, 1], *Edit_Distance* calculates the edit distance between two strings and *max_length* calculates the length of the longer string.

The second step in our algorithm attempts match concepts from two ontologies by measuring similarities between their contents. The content of a concept is a list of attributes; each attribute has a name and a data type. To compare the content for two concepts we will actually compare two lists of attributes, this comparison will proceed in two steps, firstly comparing the data type then comparing attribute name in order to find the matched attribute.

Data type comparing: In order for a pair of attributes to be matched their data type should be compatible with each other we will use data type matching in order to restrict the number of attribute to be compared. Figure 3 presents an example of data type matching.

We have two list of attributes one for product concept and the other for electronic equipments concept. If we want to match the attribute from two lists with each other without firstly comparing their data type, we will do twelve operation since product concept has three attributes and electronic equipments has four attributes, but by using data type comparing firstly we will restrict the matching operations as an example instead of comparing brand to four attribute in electronic equipment attribute list this will be restricted to compare with condition and brand since they have the same data type of brand which is string.

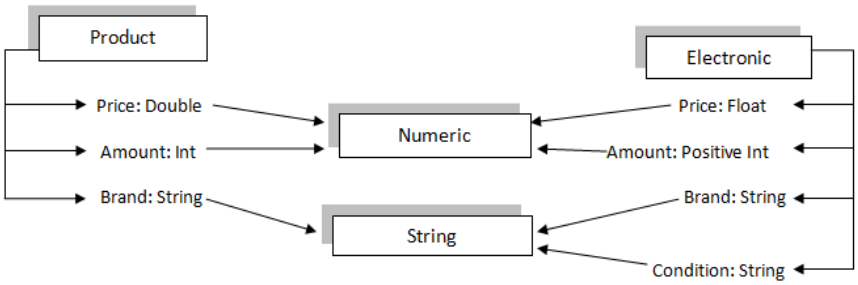

Fig. 5: Content matching based on data type abstraction

We still have a problem with data type comparison, as noticed in Fig. 3 there is more than one way to encode the same data type. For example, in product concept price is encoded as double, while in the second concept price is encoded as float. However, both attributes used for the same purpose. This different encoding for the same attribute will misled the attribute matching process since it depends firstly on data type matching. The proposed solution in order to take data types into account is to build abstractions out of the data types, Fig. 4 shows the classification of built in XML data type (Biron *et al*., 2004).
Data type will be abstracted into:

- Numeric (double, float, decimal)
- String (token, name, normalized String)
- Booleans (true/false, yes/no, 1/0)
- Date and time

Using data type abstractions will alleviate the problem that if the compared datatypses belong to the same category then they will be matched. As an example, attribute such as price will be matched from two lists if they have any numeric value, because price may be int, float, double, positiveInteger, etc. As noticed from Fig. 5, price and amount from product concept will be compared to price and amount from electronic equipments concept. Also, brand from product concept will be compared to brand and condition in the second concept since they have the same data type which is string. As a conclusion, depending on data type abstractions and by comparing the data-type for the element before the actual matching for the attribute the number of comparison is reduced to sex comparisons instead of twelve.

Attribute comparing: Until now we have defined the attributes that have compatible data types, for these attributes the next step is to find the similarities in attributes names in order to find the number of matched attributes between the compared lists, the process of computing similarities between attributes names will depend on the Eq. (1) on the first step, then if the similarity value for the compared attributes names is greater than a specific threshold which we set equal to 0.60 then the compared attributes will considered to be matched. The content similarity value between two concepts *sC* is a real value in the range [0, 1] and calculated according to the following function:

$$sC = Count/N \quad (2)$$

where,
Count = The number of pairs of properties matched
N = The length of the list with more attributes

Indirect ontology matching: Traditional ontology matching algorithms are restricted to the use of direct matching which depend on the information in the ontologies being matched, some of the new techniques (Giunchiglia *et al*., 2005; Laclavík, 2005; Małgorzata, 2009; Zharko *et al*., 2006) go beyond this and use external background knowledge in the matching. In our algorithm we will use background knowledge in the form of Word Net.

The third step in our algorithm attempts match concepts from two ontologies by discovering the relationships between the compared concepts, the relationships are discovered by using Word Net.

The relations provided by WordNet will be translated to semantic relations according to the following rules:

- A $\sqsubseteq$ B if A is a hyponym, meronym or troponym of B
- A $\sqsupseteq$ B if A is a hypernym or holonym of B
- A $\equiv$ B if they are connected by synonymy relation or they belong to one synset
- A $\perp$ B if they are connected by antonymy relation

Then each of the semantic relations will take a value in [0, 1] range according to its strength:

- $\equiv$ take the value 1
- $\sqsubseteq$, $\sqsupseteq$ take the value 0.5
- $\perp$, idk take the value 0.0

EXPERIMENT AND RESULTS

We have performed our experiment in which we matched product ontology to electronic equipments

Table 2: Name similarity

	Electronic equipment	Phone	Cell phone	Digital camera	Computer	Processer	Monitor	Computer memory
Product	0.20	0.28	0.00	0.00	0.12	0.33	0.00	0.13
Book	0.05	0.20	0.10	0.00	0.12	0.22	0.28	0.13
Text book	0.15	0.12	0.20	0.07	0.00	0.11	0.12	0.20
Magazine	0.10	0.25	0.20	0.21	0.00	0.00	0.12	0.13
Camera	0.15	0.00	0.10	0.42	0.37	0.11	0.00	0.26
Camcorder	0.15	0.22	0.10	0.21	0.44	0.22	0.22	0.33
Mobile phone	0.25	0.41	0.58	0.14	0.16	0.16	0.33	0.26
Laptop	0.20	0.16	0.30	0.14	0.25	0.11	0.28	0.20
Peripheral device	0.15	0.17	0.23	0.23	0.11	0.17	0.11	0.23
CPU	0.10	0.00	0.20	0.07	0.37	0.11	0.00	0.20
Monitor	0.15	0.14	0.10	0.21	0.37	0.33	1.00	0.26
TV	0.05	0.00	0.00	0.70	0.12	0.00	0.14	0.06
Clothing	0.20	0.25	0.40	0.70	0.12	0.11	0.12	0.13
Shoes	0.10	0.60	0.30	0.70	0.12	0.22	0.14	0.06
Running shoe	0.15	0.16	0.25	0.14	0.00	0.16	0.25	0.13
Handbag	0.05	0.14	0.00	0.14	0.00	0.00	0.14	0.00

	Computer accessory	Printer	Matrix printer	Television	Musical instruments	Guitar	Drum
Product	0.16	0.28	0.14	0.00	0.11	0.00	0.28
Book	0.16	0.00	0.00	0.10	0.00	0.00	0.00
Text book	0.16	0.00	0.14	0.10	0.05	0.12	0.00
Magazine	0.16	0.12	0.30	0.10	0.27	0.10	0.00
Camera	0.27	0.14	0.30	0.10	0.22	0.00	0.00
Camcorder	0.27	0.22	0.28	0.00	0.22	0.11	0.11
Mobile phone	0.22	0.08	0.35	0.25	0.22	0.08	0.00
Laptop	0.16	0.14	0.21	0.20	0.11	0.16	0.00
Peripheral device	0.16	0.29	0.17	0.17	0.11	0.11	0.05
CPU	0.16	0.00	0.07	0.00	0.11	0.00	0.25
Monitor	0.22	0.28	0.28	0.20	0.16	0.42	0.00
TV	0.05	0.14	0.07	0.20	0.05	0.16	0.00
Clothing	0.11	0.00	0.14	0.20	0.22	0.12	0.00
Shoes	0.16	0.14	0.07	0.20	0.11	0.00	0.00
Running shoe	0.16	0.16	0.14	0.16	0.22	0.16	0.08
Handbag	0.05	0.00	0.07	0.00	0.11	0.14	0.14

ontology by using the general matching process of our algorithm. In this section we will explain the main results from our algorithm.

Table 2 shows the similarity values between different concepts names according to Eq. (1). The *cell* [*i*, *j*] in this table stores the similarity value between the i^{th} concept in product ontology and the j^{th} concept in electronic equipment ontology.

As an example, the *cell* [7, 4] stores the name similarity value between laptop from product ontology and computer from electronic equipment ontology where:

$sN = 1 - (edit\_distance) / (max\_length)$
$edit\_distance\ (laptop, computer) = 6$
$max\_length = 8$
$sN = 1- (6/8)$ *which equals to* 0.25

The content similarity value between laptop and computer according to Eq. (2) *is*:

$sC = 8/10 = 0.8$

According to our example the overall similarity between laptop and computer is calculated as:

$Os = (w1*sN) + (w2*sC) + (w3*sR)$
$Os = (0.27*0.25) + (0.40*0.8) + (0.33*0.5)$
$Os = 0.55$

We have determined that any two concepts have an overall similarity value greater than 0.30 will be considered equivalence and so laptop and computer will be matched according to our algorithm.

CONCLUSION

We have conclude that ontologies can be used to support different tasks in multiple research areas, because it form the heart of knowledge representation for any given domain, we can say that ontology form as data source for many different interested parties. However, due to the heterogeneity among ontologies which arises when different parties design ontologies according to their own conceptual views of the domain, ontologies need to be matched before they are able to be made better for use. The objective of the work is to introduce a method for finding semantic correspondence among the ontologies with the intention to bridge communications between heterogeneous systems.

In this study we have introduced a three dimensional algorithm for ontology matching problem by comparing ontologies concepts from three dimensions:

- Matching concepts based on name similarity
- Matching concepts based on content similarity
- Matching concepts based on relationship similarity

By comparing concepts from three dimensions we can trust our result since; the same concepts may be named by different labels or they may have the same names but differ in their attribute or their relations. In these cases if we rely on one dimension in the matching process; we will not discover the similarities. On the contrary, when the matching process relies on three dimensions, if one dimension can't discover the similarity the other will do.

At the first step in our matching algorithm we have computed the similarities in names for every concept from our source and target ontology according to edit distance function. At the second step we have computed the contents similarities; this process proceeds in two steps. Firstly, comparing the data type this will restrict the number of attribute to be compared; if the data types for the compared attributes are compatible with each other then the attributes names is compared in order to find the matched attributes. At the third step, we have used Word Net as external background knowledge in order to discover the semantic relation between the compared concepts. The overall similarity for the compared concept is calculated as a weighted sum for names similarity; contents similarity and relationship similarity, respectively.

We have conducted our experiment using a real-life ontology matching scenario, we have developed our datasets from electronic market places; the data in our ontologies is collected from:

- Amazon.com
- Shopping.com
- Buy.com

Finally the results from our algorithm are summarized. We have summarized the main contribution of this study on the following:

- Building a three dimensional approach for ontology matching
- We have implement a program to learn weights for different dimension of ontologies concepts through Applying a neural network technique
- Integrating syntactic and semantic matching in our algorithm

In future study we would like to implement the three dimensional algorithm in different domains. By using datasets specialize in some domain; according to this datasets we will use an external resource that is specialize at the same domain of the dataset in order to discover more accurate relationships.

REFERENCES

Biron, P., K. Permanente and A. Malhotra, 2004. XML Schema Part 2: Datatypes. 2nd Edn., W3C Recommendation.

Chirstopher, M.D., R. Prabhakar and H. Schütze, 2009. An Introduction to Information Retrieval. Cambridge University Press, New York.

Fasli, M., 2007. Agent Technology for E-commerce. Wiley and Sons, Chichester.

Giunchiglia, F., P. Shvaiko and M. Yatskevich, 2005. Semantic schema matching. Proceedings of the 13th International Conference on Cooperative Information Systems (CoopIS 05), AgiaNapa, Cyprus.

Huang, J., J. Dang, J. Vidal and M. Huhns 2007. Ontology matching using an artificial neural network to learn weights. Proceeding of the IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition, pp: 80-85.

Laclavík, M., 2005. Ontology and agent based approach for knowledge management. Ph.D. Thesis, Institute of Informatics, Slovak Academy of Sciences.

Madhavan, J., P.A. Bernstein and E. Rahm, 2001. Generic Schema Matching with Cupid. VLDB 2001.

Madhavan, J., P.A. Bernstein, A. Doan and A. Halevy, 2005. Corpus-based schema matching. Proceedings of the 25th International Conference on Data Engineering (ICDE 05), Tokyo, Japan.

Małgorzata, M., 2009. The methodology for finding suitable ontology matching approaches. Ph.D. Thesis, Universität of Berlin.

Williams, A.B., 2004. Learning to share meaning in a multi-agent system. Auton. Agents Multi Ag. Syst., 8(2): 165-193.

Yatskevich, M. and F. Giunchiglia, 2004. Element level semantic matching. Proceeding of the Workshop on Meaning Coordination and Negotiation at ISWC, Hiroshima, Japan.

Zharko, A., K. Michel, K. Warner and H. Frank, 2006. Matching unstructured vocabularies using background ontology. Proceeding of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2006), pp: 182-197.