Knowledge Discovery through Ontology Matching: An Approach based on an Artificial Neural Network Model

M. Rubiolo, M.L. Caliusco, G. Stegmayer, M. Coronel and M. Gareli Fabrizi

Abstract

With the emergence of the Semantic Web several domain ontologies were developed, which varied not only in their structure but also in the natural language used to define them. The lack of an integrated view of all web nodes and the existence of heterogeneous domain ontologies drive new challenges in the discovery of knowledge resources which are relevant to a user's request. New approaches have recently appeared for developing web intelligence and helping users avoid irrelevant results on the web. However, there remains some work to be done. This work makes a contribution by presenting an ANN-based ontology matching model for knowledge source discovery on the Semantic Web. Experimental results obtained on a real case study have shown that this model provides satisfactory responses.

1. Introduction

Currently, the web consists of a large number of distributed and connected texts. These texts, however, are neither understood by software applications nor really used by them. The key for solving this problem is to improve both the ability of software applications to communicate directly with each other, and the representation of information in a way far more usable by them. An important framework for creating such capabilities can be provided by the next generation of the Web architecture: the *Semantic Web* [30]. The Semantic Web is intended to be a paradigm shift just as powerful as the original Web. It will bring meaning to Web page contents, in which software agents roaming from page to page can execute automated tasks by using metadata or semantic annotations, ontologies and logic [1].

Ontologies have become a de facto standard for the semantic description of data, resources and services in large distributed systems such as the Web [2]. It has been predicted that within the next decade, the Web will be composed of small highly contextualized ontologies, developed with different languages and different granularity levels [19] [20]. In this context, the simplest document will consist of concrete facts, classes, properties definitions and metadata [23]. The websites will have not only a domain ontology to describe the knowledge they can provide, but

Email addresses: CONICET, CIDISI-UTN-FRSF (M. Rubiolo, M.L. Caliusco, G. Stegmayer, M. Coronel and M. Gareli Fabrizi), mrubiolo@santafe-conicet.gov.ar (M. Rubiolo, M.L. Caliusco, G. Stegmayer, M. Coronel and M. Gareli Fabrizi)

(2011) Information Sciences, 194, pp. 107-119.

also an adequate structure to receive mobile software agents that will travel through the net (for example, looking for knowledge required by an end-user) [30].

Although the capabilities and scope of the Semantic Web are impressive today, its continuous evolution presents many problems to be faced. For instance, whenever a node on the Web needs to initiate a dynamic collaborative relationship with another, it certainly finds it difficult to know which node to contact or where to look for the required knowledge. Therefore, it can be seen that the knowledge resource discovery in such an open distributed system becomes a major challenge. This is due to the lack of an integrated view of all the available knowledge resources.

Besides, the existence of multiple domain-specific heterogeneous ontologies and their distributed development introduces another problem: on the Semantic Web, many independently developed ontologies describing the same or very similar fields of knowledge, co-exist. These ontologies are either non-identical or present minor differences, such as different naming conventions or different structures in the way they represent knowledge. Any application that involves multiple ontologies must establish semantic mappings among them to ensure interoperability. Examples of such applications arise in many domains, including e-commerce, e-learning, information extraction, bioinformatics, web services, tourism, among others [15]. For that reason, ontology-matching is necessary to solve the problem.

Ontology-matching techniques, essentially, identify semantic affinity between concepts belonging to different ontologies. Among recent proposals, machine learning methods can process the matching problem through the presentation of many correct (positive) and incorrect (negative) examples. Such algorithms require sample data from which to learn. Matchers using machine learning usually operate in two phases: (i) the learning or training phase and (ii) the classification or matching phase. During the first phase, training data for the learning process of a matcher is created, for example by manually matching two ontologies. During the second phase, the trained matcher is used for matching new ontologies. Learning can be processed on-line -the system can continuously learn- or off-line, so its speed is not as relevant as its accuracy. Usually, this process is carried out by dividing the available data set. For instance, considering a set of positive and negative examples of alignments, the examples would be divided into a training set (typically 80% of data) and a validation or test set (typically 20% of data), which would be used for evaluating the performance of the learning algorithm [17].

This paper presents a novel approach for improving web knowledge resource discovery on the Semantic Web based on recently developed intelligent techniques. This approach combines agent-based technologies with a machine-learning classifier (in particular an Artificial Neural Network (ANN) model) used for defining the matching operation between ontologies, and makes a proposal for codifying ontology knowledge as inputs to the neural model. The method proposed in this paper takes into account both schema-level and instance-level information from ontologies and semantic annotations, and proposes a machine learning solution to the ontology-matching problem, which has been proved in a real world case study, obtaining high matching rates.

The paper is organized as follows. Section 2 introduces some issues related to ontology-matching and ANN concepts. Section 3 presents related work. In section 4, a motivating scenario for knowledge discovery is introduced, and an intelligent web-agent is presented. Section 5 explains the ANN-based ontology-matching model used by this agent. Section 6 presents the model evaluation by considering a case study. Section 7 compares the ANN-based model with the H-Match ontology matching algorithm. Finally, conclusions and future work can be found in Section 8.

2. Preliminaries

The purpose of this section is to provide broad definitions of the terminology used in this work.

2.1. The Role of Ontologies and Software Agents

Ontologies provide a number of useful features for intelligent systems, as well as for knowledge representation in general, and for the knowledge engineering process [19]. In the literature, there are different definitions about what an ontology is and its different classifications [21] [20]. In this paper, a *domain ontology* is considered a representational artifact of the semantics of a given domain of discourse. A *domain* is a portion of reality that forms the subject-matter of a single science or technology [31]. The representational units of an ontology are the following: *terms*, *relations*, *properties*, *axioms* and *instances*. Since domain ontologies provide a shared conceptualization of a certain domain, they can be used for describing resource semantics by adding metadata to them.

In order to define the semantics for a digital content, it is necessary to formalize the ontologies by using specific languages such as Resource Description Framework (RDF) and Web Ontology Language (OWL) [32]. While RDF is a general-purpose language for representing information about resources in the Web, OWL is a semantic markup language for publishing and sharing ontologies. Although RDF was originally meant to represent the metadata of web resources, it can also be used to represent information about objects that can be identified on the Web. The basic construction in RDF is an *(Object, Property, Value)* triplet: a subject S has a property P with value V. An RDF-triplet corresponds to the relation that could be written as (S, P, V), for example $(http://www.books.org/ISBN0012515866, hasPrice, 62)$; and $(Professor, teachesSubject, ArtificialIntelligence)$ in the case of a University website annotated with an ontology.

Besides ontologies, software agents will play a fundamental role in building the Semantic Web of the future [23]. When data is marked up using ontologies, software agents can understand the semantics better and, therefore, locate and integrate data more intelligently, for a wide variety of tasks. A new emergent category of agents, named intelligent web (or personal) agents, would find more possible ways to meet the needs of a user and offer different choices to accomplish their goals. A personal software agent on the Semantic Web must be capable of receiving tasks and preferences from a user, seeking information from heterogeneous web sources [5].

2.2. Ontology Matching

Ontology-matching aims at finding correspondences between semantically related elements of different ontologies. The correspondences may stand for equivalence as well as other relations, such as consequence, subsumption, or disjointness, between the elements. For that reason, the problem of how to integrate heterogenous domain ontologies in the Semantic Web can be finally solved by using ontology matching [12].

In this paper, an *ontology-matching process* is formally defined as:

$$(S, P, V) \rightarrow ((O_1, c_1), \cdots, (O_i, c_i)) \tag{1}$$

where (S,P,V) represents the triplet (subject, property, value), i is the number of considered ontologies that belong to the same field of knowledge and c_i represents the result of the matching process which is a numerical matching value between 0 and 1 related to each considered ontology O_i.

There are different algorithms for implementing the matching process which can be generally classified along two dimensions [12]. On the one hand, there is a distinction between schema-based and instance-based matching. A schema-based matcher uses different aspects of the concepts and relations in the ontologies and applies some similarity measure to determine correspondence. An instance-based matcher takes the concept instances and compares them to discover similarity. On the other hand, there is a distinction between element-level and structure-level matching. Whereas an element-level matcher compares properties of a specific concept or relation to find similarities, a structure-level matcher compares the structure of the ontologies to find similarities. These matchers can also be combined [17].

2.3. Artificial Neural Networks

Artificial Neural Networks (ANNs) can be loosely defined as a large set of simple interconnected units (neurons), which are executed in parallel to perform a common global task. These units usually undergo a learning process that automatically updates network parameters (the connections among neurons, also named synaptic weights) in response to a possibly evolving input environment [8].

When the learning process is supervised, it is given a set of inputs $x = (x_1, \cdots, x_n) \in \Re^n$ and their corresponding set of target outputs $t = (t_1, \cdots, t_m) \in \Re^m$. The assumption of an ANN model is that the process that produces the output response is given by some unknown mathematical relationship $t = f(x)$, which is generally a non-linear function. Therefore, the approximation of the unknown f is performed by using a given set of training examples: some inputs x and their associated targets t. When learning is unsupervised, there are no targets. Inside the ANN model, each neuron has an activation function σ, which processes the incoming information from other neurons. Training is an optimization process where internal parameters of the neural model (the weights) are adjusted to fit the training data [22].

An ANN model can be classified according to the type of connections among their neurons. A network is named feedforward if the data flow from inputs to outputs, without feedback. This model topology, which is the most widely used, has distinct layers such as input, hidden and output, with no connections among neurons belonging to the same layer. The number of layers and neurons in each layer is chosen a-priori, as well as the type of activation functions for the neurons. An example of this kind of neural topology is the two-layer Multi-Layer Perceptron (MLP). The output of a two-layer MLP model is the following:

$$y_o = \sigma_o \left(\theta_o + \sum_{h=1}^{N_H} w_h^2 \; \sigma_h \left(\theta_h + \sum_{i=1}^{N_I} w_{h,i}^1 \; x_i \right) \right) \tag{2}$$

where N_H is the number of hidden neurons in the hidden layer and N_I is number of input variables. The o^{th} output neuron receives as input the output response of the previous hidden layer, which in this case is $\sigma_h \left(\theta_h + \sum_{i=1}^{N_I} w_{h,i}^1 \; x_i \right)$. Each of these input values is multiplied by a corresponding

weight w_h^2, and then all the values are summed. A shift θ_o (called threshold bias, an extra parameter) is applied to the above sum. Finally an activation function σ_o is applied over the whole sum. Common choices for the activation function are the sigmoid ($\sigma(x) = \frac{1}{1+e^{-x}}$), the hyperbolic tangent ($\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$) or the linear function ($y(x) = x$).

In the MLP model, learning is supervised. The basic learning algorithm used is backpropagation [22] which uses gradient descend to minimize a cost function, generally defined as the mean square error (MSE) between the desired output (targets) and the actual network output. During learning, the error propagates backwards through the network and the model parameters are changed accordingly [29].

One of the major findings with regards to MLP models is the so-called *universal approximation theorem* [11] which states that given enough neurons in the hidden layer, an MLP neural model can approximate any continuous bounded function to any specified accuracy. That is to say, a three-layer MLP neural network can approximate any arbitrary nonlinear continuous multidimensional function to any desired accuracy, provided that the model has enough neuronal units [24].

3. Related Work

In response to the challenge of ontology-matching on the Semantic Web, and in other application contexts, several proposals have appeared lately, which apply machine learning techniques to create semantic mappings. For instance, recent ANN-based tools proposed in the literature use information regarding ontology schemas and instances to produce rules for ontology integration in heterogeneous databases [17]. ANNs have been used not only for discovering correspondences among attributes via categorization and classification [27] or by learning matching parameters, such as matching weights, but also for tuning matching systems with respect to a particular matching task [16].

Given schema-level and instance-level information, it is sometimes useful to cluster these data into categories to lower the computational complexity of further data manipulations. The self-organizing map network can be used for this purpose, since the neurons in the network organize themselves according to the characteristics of given input patterns. This kind of neural model is used in the X-SOM ontology mapper [10], which uses a neural network model to weight existing matching techniques. They are combined with reasoning and local heuristics for solving semantic inconsistencies.

During the matching process, different semantic aspects such as concept names, concept properties, and concept relationships contribute in different degrees to the matching result as a vector of weights. How to learning them is not a trivial task, and current research work depends mostly on human heuristics. In [25] an ANN model learns and adjusts those weights in order to avoid some of the disadvantages of both rule-based and learning-based ontology matching approaches, which ignore the information that instance data may provide. Similarly, the work of [9] uses instances to learn similarities between ontology concepts to create then a newly combined ontology. The recent work [28] proposes an adaptive approach for finding semantic correspondences between similar elements of different ontologies. The approach first measures both linguistic and structural similarities of the ontologies in a vector space model, and then uses a neural network estimator for those similarities.

In [26] an ontology mapping strategy based on instances is presented for supporting automated

interoperability between ontology-based information systems in distributed environments. There are some methods for structurally inducing class relations from individuals. They are usually based on string-based and internal-based techniques, without considering the structure of the ontology [17]. The majority of these methods were created for two main purposes: (i) to integrate different databases, such us Automatch [3], iMAP [14] and Dumas [4]; and (ii) to merge two ontologies sharing the same set of instances, such us FCA-Merge [34]. However, most of the ontology matching tools rely on heuristics that detect some sort of similarities in the description of the concepts and the structure of the ontology graph. Since these tools generally work at the terminological level of the ontologies without taking their instances into account, they are not appropriate for the scenario described in this paper.

The H-Match [7] and GLUE [13] [15] algorithms are closely related to our model. However, the GLUE systems do not work very well if instances differ syntactically. Moreover, they cannot determine a mapping if there is a lack of instances. For that reason, we have chosen only the H-Match algorithm to compare our proposal, and the obtained results are shown in Section 7.

The model proposed in this paper is based on neural networks, taking into account schema-level and instance-level information. Our model is focused on the hypothesis that labels are human identifiers (names) for instances, which are normally shared by a community with a common language within a specific field of knowledge. Therefore, it can be stated that, if labels are the same, the instances associated with them are likely to be the same or semantically related [16]. A term in an ontology, however, can also be defined as representative of a set of instances. We can, therefore, infer that terms having the same instances are the same; and, instances having the same mother term (label) are similar. It could be assumed that, in order to perform its tasks efficiently, a search web-agent should specialize in a specific domain, in which it is expected that different ontologies will manage similar terms and contents. These facts, combined with supervised learning through examples, can be better used by an intelligent web agent to provide a more effective and efficient response to a user's search. The originality of our work relies on a novel proposal for codifying the ontology knowledge as inputs to the neural model, independently of its language; combined with an Artificial Neural Network-based model trained as a classificator for the matching of a term and an Ontology.

4. Knowledge Source Discovery on the Semantic Web

The objective of this section is to present a motivating scenario and an intelligent web agent in order to show the proposed ANN-based ontology matching model, which will be explained in detail in the following sections.

4.1. A Motivating Scenario

In open distributed systems, several nodes, probably distributed among different organizations, need resources and information (such as data, documents, services) provided by other nodes in the net. In this scenario, a key problem is the dynamic discovery of knowledge resources, namely the capacity of finding knowledge resources, from the set of available source of information, that better satisfy a given request [6]. When data is marked up using ontologies, software agents can better understand the semantics and, therefore, more intelligently locate and integrate knowledge

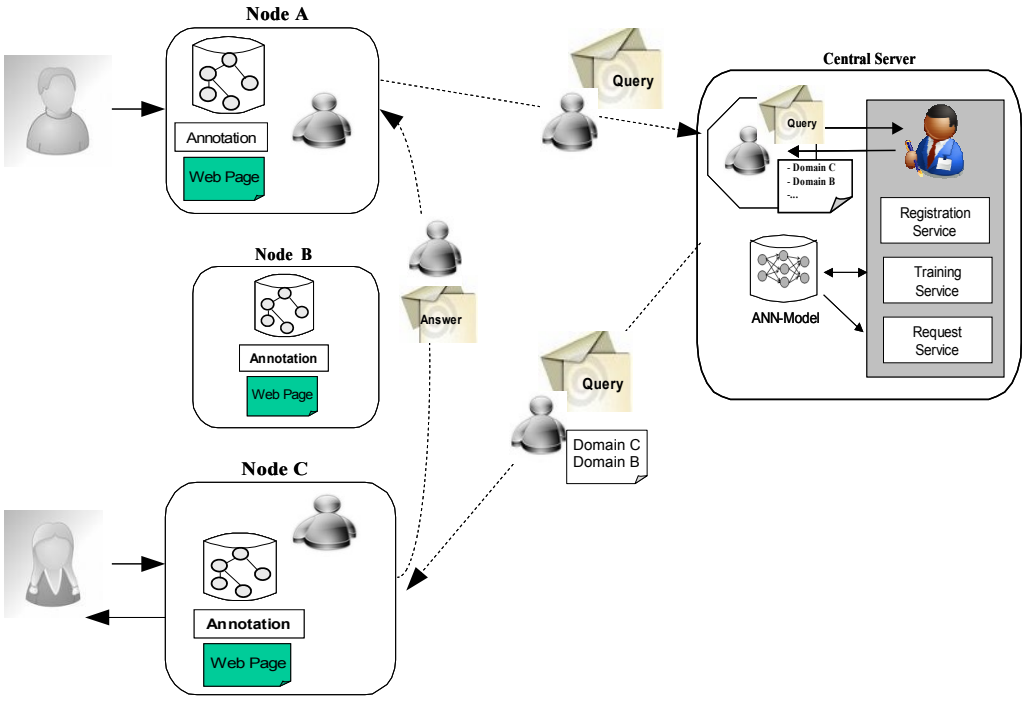

Figure 1. Reference Architecture adapted from [33].

for a wide variety of tasks.

The following example shows a motivating scenario for knowledge discovery on the Semantic Web (adapted from [33]): *A researcher from a European university wants to identify potential partners in American universities to collaborate on a project by answering the call of the European Union Framework Program - http : //cordis.europa.eu/fp7/home_en.html. The main topic of the project is the Semantic Web, a topic with which the researcher and his current colleagues are unfamiliar. A senior researcher will be required given the mandates of the project.*

In such scenario, an intelligent agent should be capable of dynamically discovering the appropriate sources of the required knowledge by matching ontologies. An agent capable of performing this task is explained in the next subsection.

4.2. The Knowledge Source Discovery Agent

In [33], an architecture for discovering knowledge sources on the Semantic Web was proposed, which is composed of mobile agents, the Knowledge Source Discovery (KSD) agent and web nodes (see figure 1). Mobile agents receive a request from the user and, using a list generated by the KSD, visit the nodes looking for an answer. The KSD agent is responsible for knowing which nodes can provide knowledge within a specific area, and for indicating a route to mobile agents that carry a user's request. The KSD agent knows the location (url) of the nodes that can provide knowledge, but it cannot actually provide the files, pictures, documents, etc., that are storaged at the nodes.

The other components of the architecture are the nodes. Each node has its own ontology used to semantically markup the information published on their websites. Suppose there are four nodes

sinc(i) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Rubiolo, M. L. Caliusco, G. Stegmayer, M. Coronel & M. Gareli Fabrizi; "Knowledge Discovery through Ontology Matching: An Approach based on an Artificial Neural Network Model"
Information Sciences, Vol. 194, pp. 107-119, 2012.

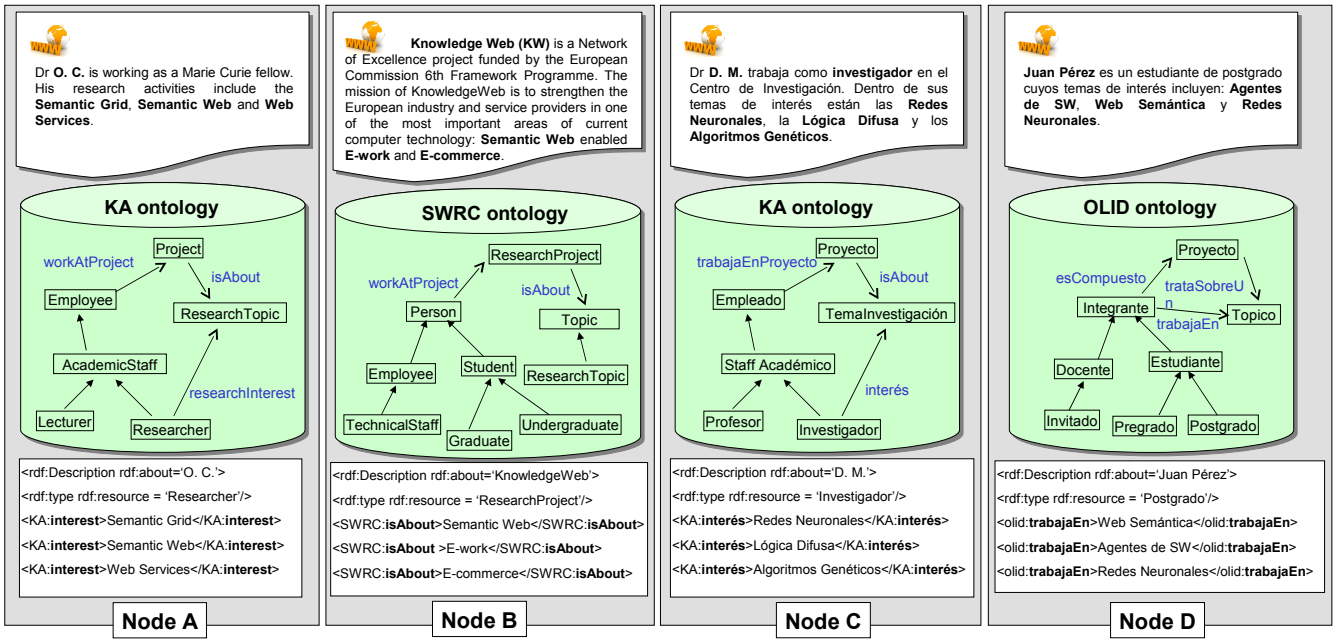

Figure 2. Domains belonging to the R+D field and their semantic annotations.

(A, B, C and D) which belong to the Research & Development (R+D) field of knowledge (Figure 2). Node A and C use the KA-ontology [1], while the node B uses the Semantic Web for Research Communities (SWRC) ontology [2], and finally, node D uses the OLID ontology[3]. All of these ontologies are implemented in OWL language. As it can be seen, each node may use a different ontology to semantically annotate the provided information, even if they belong to the same field of knowledge. Furthermore, they can vary not only in the way they represent a concept, but also in the language (for example, English, Spanish or both).

RDF is used to define an ontology-based semantic markup for each node. Each RDF-triplet assigns entities and relations to the text, which are linked to their semantic descriptions in an ontology. For example, in Node A, the following RDF-triplets: <`O.C., interest, Semantic Grid`>, <`O.C., interest, Semantic Web`> and <`O.C., interest, Web Services`> represent the O.C. research interests described in the text (see the left area of figure 2).

The KSD agent must be capable of dynamically identifying which nodes could satisfy a request brought to it by a mobile agent. This dynamic knowledge discovery requires models and techniques which allow identifiying ontology concepts that have semantic affinity among them, even when they are syntactically different. In order to do this, the KSD agent has to be able to match (probably different) domain ontologies. To face this ontology-matching problem, we propose the use of an ANN model with supervised learning stored in the KSD agent Knowledge Base (KB), which is periodically trained and re-trained off-line. This neuroal model is exolained ind etail in the next section.

[1] http://protege.cim3.net/file/pub/ontologies/ka/ka.owl
[2] http://ontoware.org/projects/swrc/
[3] http://libertad.univalle.edu/jsequeda/ontology/olid.owl

sinc(i) Research Center for Signals, Systems and Computational Intelligence (fich.unl.edu.ar/sinc)
M. Rubiolo, M. L. Caliusco, G. Stegmayer, M. Coronel & M. Gareli Fabrizi; "Knowledge Discovery through Ontology Matching: An Approach based on an Artificial Neural Network Model"
Information Sciences, Vol. 194, pp. 107-119, 2012.

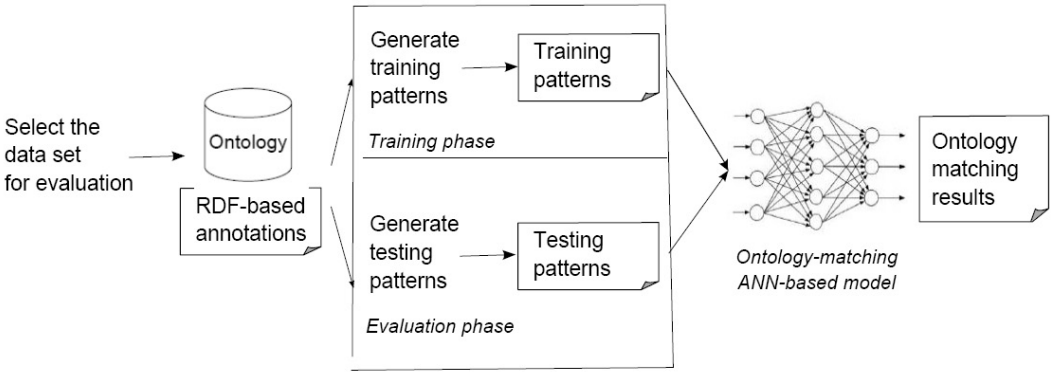

Figure 3. ANN-based Ontology Matching Model creation, training and matching phases.

The KSD agent must also be capable of understanding the natural language-based query received from the client, which is translated into an RDF-triplet (this process is out of the scope of this work). This RDF-triplet is codified before entering the ANN-based matching model. A WordNet Corpus, which could be composed of WordNet databases in different languages, is used by the KSD agent for this task. WordNet is a lexical database for the English language [18]. It groups English words into sets of synonyms called *synsets*. It is worth clarifying that while a group of synonymous words or collocations (sequences of words that together form a specific meaning) are included in one synset; different senses of one word are included in different synsets. The meaning of the synsets is further clarified by short defining glosses. Most synsets are connected to other synsets via a number of semantic relations, which vary according to the type of word, and include synonyms, among others. This research uses WordNet[4] 1.6 since different wordnets for several languages (such as Spanish[5]) are structured in the same way.

5. The ANN-based Ontology-matching Model

As stated in the introduction, machine learning matchers work in two phases: training and matching (see Figure 3). The objective of this section is to describe how the proposed neural network model is developed, trained, and used.

5.1. The ANN-based Ontology-matching Model Definition

According to the classification of matching methods presented in section 2.2, our ANN-based matcher uses schema-level information, which is extracted from the domain ontologies, as well as instance-level information which is extracted from the RDF annotations based on this domain ontology.

For neural networks, a matching problem can be viewed as a classification problem. First, the ANN-based matcher uses the schema-level information and instance-level information associated with ontology X to learn a classifier for node X, and then it uses schema-level information and

[4]http://wordnet.princeton.edu/
[5]http://www.lsi.upc.edu/nlp

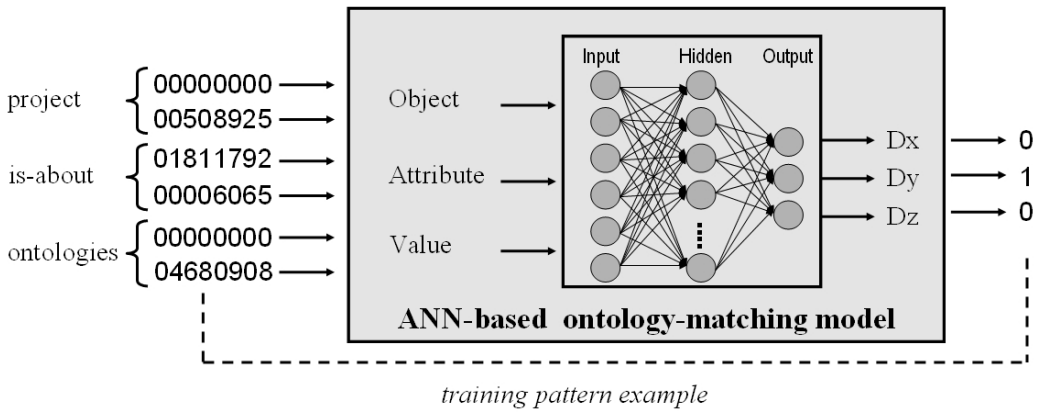

Figure 4. ANN-based ontology-matching model and a training pattern example.

instance-level information associated with ontology Y to learn a classifier for node Y. After that, it classifies elements of Y according to the X classifier, and vice-versa. Hence, we have a method for identifying elements of $X \bigcap Y$. The same idea is applied to more than two nodes.

In order to build a classifier for node X, triplets must be extracted from its domain ontology and from the RDF-based annotations based on this ontology. Each part of the triplet corresponds to an input unit for the neural model.

The proposed neural classification model is a multilayer perceptron (MLP). It has inputs corresponding to each triplet component. Each triplet component consists of two words. Therefore, 6 input units are considered for the MLP model. The number of hidden units is determined by a trial-and-error procedure.

The are as many outputs neurons in the model as nodes to be matched. The activation of a neuron consists of producing a value of (near) 1 when the input triplet exists in the corresponding node, and 0 in other cases.

5.2. Training Phase

In order to decide on the MLP parameters, several architectures have been tested. After 500 training epochs, the classifier topology used resulted in 12 hidden neurons with sigmoid activation functions. In these models, a different number of hidden neurons were considered and performance was tested on validation patterns, using the k-fold cross-validation methodology, (k=3, 3 repetitions per fold) combined with the Levenberg-Marquardt training algorithm [22].

During the supervised training, {input/target} pairs (called *training patterns*) are shown to the model as:

```
InputPattern = <Object; Attribute; Value>
```

with its corresponding target, indicating to which node this input pattern belongs to:

```
TargetPattern = <Dx; Dy; Dz> .
```

In order to train the ANN-based ontology matching model, these training examples must be created. Each example must tell the network that a certain triplet can be found in a certain node.

For example, let us consider the ontologies and annotations shown in Figure 2. A training pattern indicating that the triplet <*project; is-about; ontologies*> can be found in the Node B ontology, but not in A nor C, must have the form:

```
InputPattern=<project;is-about;ontologies>
TargetPattern=<0;1;0>
```

This means that, given the fact that there are projects in Node B that have research interest in ontologies, its corresponding triplet would be `<project; is-about; ontologies>` and its corresponding output target would be `<0; 1; 0>`. Only the second vector value (that represents Node B) is equal to 1, indicating that this triplet can be found in Node B.

Once the triplets for each node are created, they have to be converted from strings into numbers. We propose to code the terms triplet using the WordNet database, in which an element name is associated with a code named *synset offset*. This code is represented by an 8 digit decimal integer. In this way, an appropriate pattern-codification schema can be achieved because all terms can be codified with an invariant-length code. However, most of the terms represented in WordNet are not collocations but single words. For example, in English WordNet 1.6 the term *Semantic Web* is not a collocation. This is a problem that is addressed by assuming the collocation as two independent words. Then, a triplet term can be represented as a pair of codes, whose values will vary depending on whether the term has:

i) *a single word:* the term code will be formed by the *synset* code associated with the word, including an 8-zero-code in the first position, representing the absence of another word;
ii) *two words:* the term code will be formed by the composition of a *synset* code associated with each word.

Figure 4 shows also the triplet codification. It can be seen that for *project* the code is formed by a combination of zero and `00508925`: <`00000000;00508925`>. The code related to *is-about* is formed by *is* code `01811792` and *about* code `00006065`:<`01811792;00006065`>. A single word codification is applied to the word *ontologies*.

In summary, this training pattern would be:

```
InputPattern = <<00000000;00508925>; <01811792;00006065>; <00000000;04680908>>
TargetPattern = <0;1;0>
```

An interesting fact related to the use of different language ontologies arises from using the WordNet database for triplet term codification: the process of identifying the right node for a particular triplet could be significantly improved if all the words and their translations are codified with the same code. There is some sort of automatic triplet expansion and translation as a consequence of using the WordNet codification schema for an ANN model training. That is to say, a term in a triplet has the same code in English WordNet and in Spanish WordNet. For example, the English term *project* and its Spanish translation *proyecto* have both the same code: `00508925`. Using this unique code, it is possible to consult all nodes without taking each node language into account. Similar conclusions can be drawn in the case of synonyms. Therefore,language is not a barrier for our ontology-matching proposal.

5.3. Matching Phase

As stated before, the ANN model is trained the model will always provide a response. Besides standard learning algorithms, such as the Levenberg-Marquardt algorithm [22], can be used for model training, in which stability and convergence are guaranteed. If the request contains concepts totally unknown to the model, it will show this fact by providing output values that are very different from the training patterns.

6. ANN-based Ontology Matching Model Evaluation

Since the first step of an evaluation is to select the data set used for performing it, finding a suitable semantic data set was crucial because of the differences in form and quality that might exist among them. In fact, ontology datasets suitable for matching are not easy to find. The first problem is that public and well-designed ontologies with meaningful overlap are required. The data sets made for OAEI [6] (Ontology Alignment Evaluation Initiative) campaign can be considered correct (as far as construction is concerned) but they are neither realistic nor very hard [17]. What is more, all data sets defined for evaluating the matching algorithm are composed of a pair of ontologies. Instead, to evaluate our proposed method, more than two annotated ontologies are required.

In order to train the ANN-based ontology-matching model proposed in this work, ontologies as well as RDF-based annotations are required. Thus, we have decided to carry out an application-specific evaluation which has two main advantages: (i) it is more realistic than artificial test beds, (ii) it provides very specific information. This evaluation was divided in three phases:

1. A dataset for evaluation was built.
2. The ANN-based Model of the KSD Agent was trained and validated.
3. Different queries were made to test the ANN-based Model.

6.1. Material and methods: building the Dataset for Evaluation

This application example is based on the PAE-CELTIC Project[7]. This project actually involves six research groups, four academic institutions, six enterprises and two non-governmental organizations. These entities are located in different cities of two different provinces in Argentina. In addition, more than 100 people work on the project in different areas of Information Technology and Communication. As a result, not all of the participants in the project are acquainted with the research areas of each other. Moreover, the project structure can change over time and new entities can be added, which means having new people involved in it with new working areas.

In this dynamic scenario, it would be useful to implement the architecture presented in this work. Such architecture would facilitate searching for people and research groups who work in the same area of interest, or contacting, in advance, people who work in a particular area and were previously unknown to the end user.

With the aim of obtaining a set of RDF-based annotations for training the ANN classifier, the websites of four research groups involved in the PAE-CELTIC project were selected and semantically annotated using three different ontologies: KA, SWRC and OLID (previously described in figure 2). Table 1 shows the selected websites and the ontology that were used to annotate them. Note that in the case of nodes B and C, the language of the website and the ontology are not the same. This fact can certainly make the process of matching more difficult when different languages are mixed up in the annotations.

Using the RDQL query language[8], a set of triplets from each ontology and the semantic annotations for each node were obtained. This set was pre-processed deleting empty and incomplete triplets, such as annotations having at least an empty element because they do not provide useful information. The final set of triplets has 159 data points in total, composed of 53 annotations for each domain.

[6] http://oaei.ontologymatching.org/
[7] http://www.celtic.santafe-conicet.gov.ar/pae
[8] http://www.w3.org/Submission/RDQL/

Table 1. Nodes and domain ontologies used to annotate them

Node	*Website*	*Language*	*Ontology*
A:CIDISI	http://cidisi.frsf.utn.edu.ar	English	KA
B:INGAR	http://www.ingar.santafe-conicet.gov.ar	English	SWRC
C:SINC	http://fich.unl.edu.ar/sinc/	English	KA
D:CIMEC	http://www.cimec.org.ar	Spanish	OLID

6.2. Training the ANN-based model

The MLP model parameters were set according to typical values. The model parameters (weights and biases) were randomly initialized between -1 and 1 at the beginning of the training phase. Each neuron in the hidden and output layer was activated with a sigmoid function. The training algorithm used was standard backpropagation. The training data were normalized into the activation function node of the hidden neurons, before entering the model, since this significantly improves training time and ontology-matching model accuracy. The number of input neurons for the MLP model was set to 6, considering a double-code for each triplet term. The hidden layer neuron number was set empirically, according to the training data and the desired accuracy for the matching. The output has a specialized output neuron in the model for each node (in our case, since there are four nodes there are 4 output neurons). The allowed values for each output neuron were 1 or 0, depending on whether the neuron recognizes or not an element belonging to the node it represents.

The k-fold cross validation method was used for training [22]. The complete dataset was randomly split into $k = 3$ mutually exclusive subsets of equal size, repeating the experiments three times in each fold. The cross validation estimate of the overall accuracy of a model was calculated by simply averaging the accuracy measures over the three validation datasets. There were 53 triplets for each node, 43 were used for training and 10 for validation.

Two strategies were used for model training:

1. *Random order training*: the training triplets (belonging and not belonging to the class) are presented, during training, in random order to the model.
2. *First one known, then three unkowns (1 kn, 3 unkn)*: the neural model receives the training triplets belonging to each class in the following order: one triplet belonging to a node and three triplets not belonging to that node.

6.3. Validation results of the proposed ANN-based model

In classification problems, the primary source of performance measurements is the overall accuracy of a classifier estimated through the classification or recognition rate R. This rate is calculated as the number of correct classifications over the total number of examples to be classified.

Table 2 presents the results obtained for the ANN-based model classification rate (R) for each class $i = A, \cdots, D$, as the average over the validation set (10 triplets for each class) for all the k-folds ($k = 3$). The results show that, on average, the proposed ANN-based model for matching can achieve high classification rates of almost 80% in each class and on average, considering the average results over all the classes.

Table 2. Recognition rates (R) for the ANN-based matching model on the validation set.

	ANN random training	ANN (1 kn, 3 unkn) training
R_A	80.00%	83.33%
R_B	76.67%	76.67%
R_C	83.33%	86.67%
R_D	76.67%	63.33%
$R_{average}$	79.17%	77.50%

Table 3. Ontology-matching query examples

Query triplets	Matching Node: original annotation
1)	C:<Image Processing; type; Research Topic>,
2)	C:<process Control; type; theme>
3)	B:<Paradigma Objeto Relacional; type; Publication>
4)	B:<Paradigma Objeto Relacional; type; Publication>
5)	A:<project,is-about,ontologies>
6)	A:<fellow,interest,semanticWeb>
7)	A:<projecto, sobre, ontologías>
8)	A:<becario, interés, Web Semántica>

With respect to the influence of the order of the training patterns, two training strategies (random training vs. (1 kn, 3 unkn) training) were used to check the influence of the order of the training samples over the model classification capacity or recognition rate. As the Table 2 shows, there is an important difference between the random training schema and the ordered training one. Looking at the classification rates for each domain in detail, it can be seen that the worst rate reached over the validation set obtained for node D is, however, higher than 60%. This rate reaches a maximum of approximately 90% for node C when the classifier was trained with ordered samples for this class. The random training strategy provided the highest results in average for all classes, of almost 80%.

7. Performance comparison between the ANN-based model with an Ontology Matching Algorithm

We have compared our MLP-based ontology-matching model with the H-Match algorithm [7], a well-known algorithm for matching populated ontologies by evaluating the semantic affinity between two concepts and considering both their linguistic and contextual affinity. In order to use this algorithm, a probe query (one word) is sent to each node. Each node uses the H-Match algorithm to determine whether it has concepts inside its populated ontology that match the query.

7.1. H-Match algorithm output evaluation

The H-Match algorithm provides a semantic affinity value ($s_{t_i,n}$) for each triplet-term $< t_i >$ compared with each node ontology n, for $i, n = 1, 2, 3$. The $s_{t_i,n}$ values are combined to obtain an average matching measurement ($Av_{(<t_1,t_2,t_3>,n)}$) for the complete triplet $< t_1, t_2, t_3 >$ against a node ontology, according to the following formula:

$$Av_{(<t_1,t_2,t_3>,n)} = \frac{s_{t_1,n} + s_{t_2,n} + s_{t_3,n}}{3} \tag{3}$$

To determine if the triplet can be indicated as belonging or related to the analyzed node n, the semantic affinity measurement $Av_{(<t_1,t_2,t_3>,n)}$, as well as two of the semantic affinity values, have to be higher than an empirically set threshold α . If both conditions are satisfied, the triplet $< t_1, t_2, t_3 >$ is considered to be *matched* to the node ontology n.

7.2. ANN-based matching model output evaluation

For the ANN-based matching model proposed in this paper, the evaluation of the test set must be properly defined because a query triplet $t_i =< t_1, t_2, t_3 >$ may be included in more than one node ontology. That is to say, in this classification problem there are many patterns which have overlapping classes. In order to decide if the result provided by the MLP model is correct or not, the expected target for each test case was determined.

The targets for the test were set in the following way: a triplet t_i is defined as target for an n node if:

- t_i belongs to the n node ontology, or
- t_i does not belong to the n node ontology but it is included in n' and $o(t, n) \geq \alpha\ o(t, n')$

where $o(t_i, n)$ is the sum of the occurrencies of no-blank words in t_i in the node n triplets used in the training set, and α is a real positive number that maximizes R for the ANN-based model.

According to the previous definition of the targets, the outputs of the ANN-based matching model were evaluated: a triplet t_i is considered as belonging to an n node ontology if:

- t_i belongs to the n node ontology and $y(t, n) \geq \min(n)$, or
- t_i does not belong to the n node ontology but to n' ontology and $y(t, n) \geq \min(n')$

where $y(t, n)$ is the output of the ANN-based model neuron associated with/corresponding to the node n, and $\min(n)$ is the minimum output value obtained for an output neuron during training with the training set corresponding to the node n. This means that during training, minimum activation thresholds were identified for each output neuron that represented a class. When a test pattern was presented to the MLP model, the output of the model was evaluated considering the neurons that presented an output with a higher value than its corresponding threshold. This way, more than one neuron (class) was considered as the response to a validation (or test) query triplet.

7.3. Matching evaluation on test queries

With the aim of evaluating the performance of the ANN-based ontology-matching model and the H-Match algorithm, eight query triplets have been used for testing. The query triplets are the following:

1) *<image Processing, type, research Topic>*

Table 4. Ontology-matching results for ANN-based model (random training) vs. H-Match algorithm ($\alpha = 0.6$) on test set.

	NODE A		NODE B		NODE C		MODEL OUTPUT	
Query	ANN	H-Match	ANN	H-Match	ANN	H-Match	ANN	H-Match
1)	$(A, 0.023)$	$(A, 0.733)$	$(B, 0.161)$	$(B, 0.702)$	$(C, 0.750)$	$(C, 0.733)$	C	C, A, B
2)	$(A, 0.000)$	$(A, 0.677)$	$(B, 0.000)$	$(B, 0.682)$	$(C, 1.000)$	$(C, 0.677)$	C	B, C, A
3)	$(A, 0.037)$	$(A, 0.643)$	$(B, 0.987)$	$(B, 0.597)$	$(C, 0.000)$	$(C, 0.643)$	B	A, C
4)	$(A, 0.029)$	$(A, 0.679)$	$(B, 0.990)$	$(B, 0.637)$	$(C, 0.000)$	$(C, 0.679)$	B	A, C, B
5)	$(A, 0.750)$	$(A, 0.905)$	$(B, 0.000)$	$(B, 0.826)$	$(C, 0.000)$	$(C, 0.905)$	A	A, C, B
6)	$(A, 0.750)$	$(A, 0.640)$	$(B, 0.000)$	$(B, 0.604)$	$(C, 0.000)$	$(C, 0.640)$	A	A, C, B
7)	$(A, 0.750)$	$(A, 0.835)$	$(B, 0.000)$	$(B, 0.790)$	$(C, 0.000)$	$(C, 0.835)$	A	A, C, B
8)	$(A, 0.750)$	$(A, 0.616)$	$(B, 0.000)$	$(B, 0.587)$	$(C, 0.000)$	$(C, 0.616)$	A	A, C

2) <*process Control, type, theme*>
3) <*grupo, trabaja, objeto Relacional*>
4) <*persona, trabaja, objeto Relacional*>
5) <*project, is About, ontologies*>
6) <*fellow, interest, Semantic Web*>
7) <*projecto, sobre, ontologías*>
8) <*becario, interés, Web Semántica*>

The following criteria were used to select the test query triplets. Triplets 1) and 2) exist in the node C. Triplets 3) and 4) are similar, having only one difference in the first term. Both are in spanish and can be found in the node B. It is worth mentioning that all of these triplets are expressed in the same language that the node which should be indicated as the one containing the query triplet or a similar one in the corresponding annotated ontology. Triplets from 5) to 8) have been selected in order to prove the language influence in the ontology matching model. Triplets 5) and 7) have exactly the same terms but are expressed in english and spanish, respectively. The same occurs with triplets 6) and 8). All of their correspondences are presented in Table 3, which shows, in the first column, the query considered in the test, while the second column indicates the corresponding node containing the query triplet or a similar one.

7.4. Results and Discussions

This subsection presents the results obtained when comparing the proposed ANN-based matching model with the H-Match algorithm, using the test queries previously explained.

Table 4 shows, in the first column, the number of the triplet query considered in the test; from the second to the sixth columns, the ontology-matching results for each annotated node considered in this study (A, B and C) are presented; and the last columns report each model global output: the node ontology that is indicated as the one that could potentially respond the query. This response is related to the criteria in which a node is considered if the ontology-matching value is higher than a threshold of 0.6, because the output range is {0-1}. When more than one node fullfills this requirement, the model

output is presented as a ranking of potential nodes ordered from higher to lower matching value. The numerical values that support each model output are highlighted in the table using different colors: blue for ANN model and cyan for H-Match model.

The results reported for the ANN model correspond to a training phase using a random sampling strategy, which has provided the higher classification rates for this problem, as shown in the Table 2. In order to obtain the H-match model results, the testing examples are evaluated node by node setting the algorithm parameters as follows: matching model = *intensive*, mapping = *one-to-one*, adopts inheritance = *false*, empty context strategy = *pessimistic*, matching strategy = *standard (asymmetric)* and weight linguistic affinity = *1.0*.

The analysis of these results indicates that the ANN model matches all of the test queries; it means that in 100% of the test cases, this model correctly indicates a node that can potentially answer the query. For the H-match algorithm, seven of the eight test queries are correctly matched (87.5%). It is important to highlight that the neural model always indicates only one node. While in most other cases, the H-match model output indicates more than one possible node to answer the query.

Recalling the motivating scenario for this work, and the reference architecture presented in Figure 1, one must remember that the ontology-matching model is a component of the KSD agent, which receives queries transported by mobile agents looking for answers to them. When a list of nodes are indicated as potential responses to the query, all of them should be visited in the received order, most likely wasting time and overloading the network. This is definitely an important advantage of the ANN model over the H-match model because the former always indicates just one node.

A detailed analysis has been performed by evaluating each test query. With regard to the H-match model, triplets 1), 5), 6), 7) and 8) are considered correct because the first indicated node is the right one. For the second and fourth triplets, the output is accepted as correct although the right node does not appear in the first place. In the test triplet 3) the output is considered wrong because the node that should have been indicated as provider of an answer does not even appear in the ranking. It must be stated that due to the fact that the H-match algorithm gives more importance to the ontology structure than to its instances, this algorithm has always provided the same answers for the nodes annotated with the same ontology.

In summary, it can be stated that, for all the tests, the ANN-based ontology-matching model has provided satisfactory results. Judging from the obtained results, the proposed model is more accurate when choosing potential nodes that respond to a query, compared to a traditional matching algorithm. Note that with respect to triplets 5) to 8), these are translations of the same words, and the ANN-based model provides the same correct answer for all of the cases. This is an important characteristic of our proposed model because the combination of using a codification scheme for words (such as Wordnet) and a neural network-based ontology-matching model has proved to be independent of the language. This is an important advantage when the query language is different from the language of the annotations of the node ontology.

8. Conclusions and Future Work

This paper has presented the advantages of using an ANN classifier model inside a Knowledge Source Discovery agent for ontology-matching on the Semantic Web. As it has been accurately shown, this novel approach helps users avoid irrelevant search results and wrong decision making on the web in the future. Unlike other proposals, this model is based on a neural network that takes into account schema-level and instance-level information for training a classifier for each annotated node in the net. In addition, the language in which the ontology is expressed could be different from the language in which the website is annotated.

Through the obtained results for a case study involving three nodes belonging to the research and development domain, it can be stated that such a model is capable of learning from several heterogeneous ontologies labels and instances. Several matching request triplets have been used to test the proposed model. In all of the cases, the ANN-based ontology-matching model has provided an accurate answer. The experimental results have shown the effectiveness and efficiency of the proposed method, when compared to other state-of-the-art matching algorithms.

This work has evaluated a classic configuration for the multilayer perceptron with one output unit for each class (ontology domain) to match, since this topology is the common choice for neural classifiers. With respect to its scalability, in this approach, if the number of domain ontologies to match increases, the model has to be changed and re-trained, by simply adding more output neurons to the model. A more flexible and scalable architecture for the neural matcher would be to have a neural model specialized in each ontology domain. Such alternative topology will be the subject of future research.

Future work will also be devoted to consider a more complete case study, involving more nodes and instances. Furthermore, a more complex set of test queries should be used considering triplets which match only part of their terms with the annotated ontologies. Another important case to be tested is when a query triplet could be answered by more than one node.

References

[1] H.P. Alesso, C.F. Smith, Thinking on the Web - Berners-Lee, Godel, and Turing, A John Wiley Sons, INC, 2006.

[2] C. Badica, G. Mangioni, S. Rahimi, Intelligent distributed information systems, Information Sciences - Special Issue on Intelligent Distributed Information Systems 180 (2010) 1779–1780.

[3] J. Berlin, A. Motro, Database schema matching using machine learning with feature selection, 14th International Conference on Advanced science, volume LNCS 2348, pp. 452–466.

[4] A. Bilke, F. Naumann, Schema matching using duplicates, 21st International Conference on Data Engineering, pp. 69–80.

[5] K. Breitman, M. Casanova, W. Truszkowski, Semantic Web: Concepts, Technologies and Applications, Springer-Verlag, London, 2007.

[6] S. Castano, A. Ferrara, S. Montanelli, Dynamic Knowledge Discovery in Open, Distributed and Multi-Ontology Systems: Techniques and Applications, in: Web Semantics and Ontology, Idea Group, London, 2006.

[7] S. Castano, A. Ferrara, S. Montanelli, Matching ontologies in open networked systems: Techniques and applications, Journal on Data Semantics (JoDS) 3680 (2006) 25–63.

[8] D. Chaturvedi, Soft Computing Techniques - Studies in Computational Intelligence, Springer-Verlag, London, 2008.

[9] A. Chortaras, G. Stamou, A. Stafylopatis, Learning ontology alignments using recursive neural networks, Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, volume LNCS 3697, pp. 811–816.

[10] C. Curino, G. Orsi, L. Tanca, X-SOM: A flexible ontology mapper, DEXA Workshops, volume 1, pp. 424–428.

[11] G. Cybenko, Neural networks in computational science and engineering, IEEE Computational Science and Eng. 3 (1996) 36–42.

[12] J. Davies, R. Studer, P. Warren, Semantic Web Technologies: trends and research in ontology-based systems, John Wiley, London, 2007.

[13] A. Doan, P. Domingos, A. Halevy, Learning to match the schemas of data sources: A multistrategy approach, Machine Learning 50 (2003) 279–301.

[14] A. Doan, A.Y. Halevy, Semantic integration research in the database community: A brief survey, AI Magazine 26 (2005) 83–94.

[15] A. Doan, J. Madhavan, P. Domingos, A. Halevy, Ontology Matching: A Machine Learning Approach. Handbook on Ontologies in Information Systems, Springer, New York, 2004.

[16] M. Ehrig, Ontology Alignment: Bridging the Semantic Gap, Springer, London, 2007.

[17] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, London, 2007.

[18] C. Fellbaum, WordNet: An Electronic Lexical Database (Language, Speech, and Communication), The MIT Press, 1998.

[19] A. Gomez-Perez, O. Corcho, M. Fernandez-Lopez, Ontological Engineering, Springer, London, 2005.

[20] T. Gruber, A translation approach to portable ontology specifications, Technical Report 1 (1993) 71–92.

[21] N. Guarino, P. Giaretta, Ontologies and knowledge bases: Towards a terminological clarification, Technical Report 1 (1995) 71–92.

[22] S. Haykin, Neural Networks, MacMillan, New York, 2005.

[23] J. Hendler, Agents and the semantic web, IEEE Intelligent Systems 16 (2001) 30–37.

[24] K. Hornik, M. Sthincombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.

[25] J. Huang, J. Dang, W. Zheng, M. Huhns, Use artificial neural network to align biological ontologies, BMC Genomics 9 (20078) 1–12.

[26] J. Jung, Reusing ontology mappings for query routing in semantic peer-to-peer environment, Information Sciences 180 (2010) 3248–3257.

[27] W. Li, C. Clifton, Semint: A system prototype for semantic integration in heterogeneous databases, In Proc. of the 1995 ACM SIGMOD International Conference on Management of Data, volume 1, pp. 484–485.

[28] M. Mao, Y. Peng, M. Spring, An adaptive ontology mapping approach with neural network based constraint satisfaction, Journal of Web Semantics 8 (2010) 14–25.

[29] D. Rumelhart, G. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.

[30] N. Shadbolt, W. Hall, T. Berners-Lee, The semantic web revisted, IEEE Intelligent Systems 21 (2006) 96–101.

[31] B. Smith, W. Kusnierczyk, D. Schober, W. Ceusters, Towards a Reference Terminology for Ontology Research and Development in the Biomedical Domain. In: Biomedical Ontology in Action, 2006.

[32] M. Smith, C. Welty, D. McGuiness, OWL web ontology language guide, Recommendation W3C 2 (2004).

[33] G. Stegmayer, M. Caliusco, O. Chiotti, M. Galli, NN-agent for distributed knowledge source discovery, Lecture Notes on Computer Science 4805 1 (2007) 467–476.

[34] G. Stumme, A. Maedche, Fca-merge: Bottom-up merging of ontologies, 17th International Joing Conference on Artificial Intelligence - IJCAI 2001, pp. 225–234.