Chapter 2
Semantic Web Technologies and Artificial Neural Networks for Intelligent Web Knowledge Source Discovery

M.L. Caliusco and G. Stegmayer

Abstract This chapter is focused on presenting new and recent techniques, such as the combination of agent-based technologies and Artificial Neural Network (ANN) models that can be used for intelligent web knowledge source discovery in the new and emergent Semantic Web.

The purpose of the Semantic Web is to introduce semantic content in the huge amount of unstructured or semi-structured information sources available on the web by using ontologies. An ontology provides a vocabulary about concepts and their relationships within a domain, the activities taking place in that domain, and the theories and elementary principles governing that domain. The lack of an integrated view of all sources and the existence of heterogeneous domain ontologies, drives new challenges in the discovery of knowledge sources relevant to a user request. New efficient techniques and approaches for developing web intelligence are presented in this chapter, to help users avoid irrelevant web search results and wrong decision making.

In summary, the contributions of this chapter are twofold:

1. The benefits of combining Artificial Neural Networks with Semantic Web Technologies are discussed.
2. An Artificial Neural Network-based intelligent agent with capabilities for discovering distributed knowledge sources is presented.

2.1 Introduction

The web grows and evolves at a fast speed, imposing scalability and relevance problems to web search engines. Moreover, another ingredient is being recently added to it: data semantics. The new *Semantic Web* allows searching not only information but also knowledge. Its main purpose is introducing structure and semantic content in

M.L. Caliusco (✉)
CONICET, CIDISI-UTN-FRSF, Lavaise 610, Santa Fe, Argentina
e-mail: mcaliusc@frsf.utn.edu.ar

Y. Badr et al. (eds.) *Emergent Web Intelligence: Advanced Semantic Technologies,*
Advanced Information and Knowledge Processing,
DOI 10.1007/978-1-84996-077-9_2,

the huge amount of unstructured or semi-structured distributed knowledge available on the Web, being the central notion behind the Semantic Web that of ontologies, which describe concepts and their relations in a particular field of knowledge [1].

The knowledge source discovery task in such an open distributed system is a new challenge because of the lack of an integrated view of all the available knowledge sources. Therefore, if a part of the system wants to initiate, for example, a dynamic collaborative relationship with another one, it is difficult to know who to contact and where to go and look for the required knowledge. Besides that, the distributed development of domain-specific ontologies introduces another problem: in the Semantic Web many independently developed ontologies, describing the same or very similar fields of knowledge, co-exist. Those ontologies are not identical or present minor differences, such as different naming conventions, or higher level differences in their structure and in the way they represent knowledge. These problems can be caused, among other things, by the use of different natural languages, e.g. Paper vs. Artículo, different technical sublanguages, e.g. Paper vs. Memo, or use of synonyms, e.g. Paper vs. Article. Therefore, ontology-matching techniques are needed, that is to say, semantic affinity must be identified between concepts that, appearing in different ontologies, are related.

The web of the future will be composed by small highly contextualized ontologies, developed with different languages and different granularity levels. A simpler document in the Semantic Web will be composed by the website, the metadata that describe it and a domain ontology that represents the metadata semantics [16]. The websites will have, besides an ontology domain to describe the knowledge they can provide, an adequate structure to receive mobile software agents (i.e. an agent server) that will travel the net, for example looking for knowledge required by an end-user [2]. Considering that these agents will have their own ontology for communication, in this future context the main challenge will be how to go to the most relevant sites (avoiding waste of time) and how to communicate with them.

In addition, there must be a mechanism for defining ontology-matching, a key issue that agents have to deal with in the Semantic Web. Virtually 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, knowledge management, e-learning, information extraction, bioinformatics, web services, and tourism, among others [9].

This chapter is focused on presenting new and recent techniques that can be used for improving web knowledge source discovery in the Semantic Web, such as the combination of agent-based technologies and Artificial Neural Network (ANN) models, which can be used for defining the matching operation between ontologies.

The structure of this chapter is the following: Sect. 2.2 introduces some basic concepts regarding the Semantic Web, Ontologies, Software Agents, and Artificial Neural Networks (ANNs). Section 2.3 discusses related work with knowledge discovery on the web. In Sect. 2.4, the web knowledge discovery task is explained in detail and some motivating scenarios are introduced. Section 2.5 presents a proposal for an ANN-based ontology-matching model inside a knowledge source discovery agent and a related performance evaluation. Finally, Sect. 2.6 presents the conclusions of this chapter.

2.2 Foundations

Semantic Web uses ontologies and a number of standard markup languages to formally model information represented in web resources so that it is accessible to humans and machines working co-operatively, perhaps with the assistance of intelligent services such as software agents, which use artificial intelligent (AI) techniques and models.

The purpose of this section is to provide an introduction to some concepts that are used along the chapter, regarding Ontologies, Ontology-matching, Software Agents and Artificial Neural Networks.

2.2.1 Ontologies and Ontology-Matching

Since ontologies have been used for different purposes in different disciplines, there are different definitions about what an ontology is, most of them contradictories [13].

In this chapter, an ontology is considered as an explicit representation of a shared understanding of the important concepts in some domain of interest. Ontologies usually are referred as a graph structure consisting of [8]:

1. a set of concepts (vertices in a graph),
2. a set of relationships connecting concepts (directed edges in a graph), and
3. a set of instances assigned to a particular concept (data records assigned to concepts or relations).

In order to define the semantics for digital content, it is necessary to formalize the ontologies by using specific languages as Resource Description Framework (RDF) and Web Ontology Language (OWL). On the one hand, RDF is a general-purpose language for representing information about resources in the Web. It is particularly intended for representing metadata about web resources, but it can also be used to represent information about objects that can be identified on the Web. On the other hand, OWL describes classes, properties, and relations among these conceptual objects in a way that facilitates machine interoperability of web content[3].

Ontologies provide a number of useful features for intelligent systems, as well as for knowledge representation in general and for the knowledge engineering process. However, in open or evolving systems, such as the Semantic Web, different parties could, in general, adopt different ontologies. Thus, merely using ontologies does not reduce heterogeneity: it raises heterogeneity problems at a higher level. Ontology-matching is a plausible solution to the semantic heterogeneity problem [8].

Ontology-matching aims at finding correspondences between semantically related entities of different ontologies. These correspondences may stand for equivalence as well as other relations, such as consequence, subsumption, or disjointness, between ontology entities.

Ontology-matching results, called alignments, can thus express the relations between the ontologies under consideration with various degrees of precision [12].

Alignments can be used for various tasks, such as ontology merging, query answering, data translation or Semantic Web browsing.

Technically, *ontology-matching*, can be defined as follows [12]:

> *The matching process can be seen as a function f which, from a pair of ontologies to match O_A and O_B, an input alignment A_1, a set of parameters p and a set of oracles and resources r, returns an alignment A_2 between these ontologies.*

An alignment can be defined as:

> *Given two ontologies O_A and O_B, an alignment is made up of a set of correspondences between pairs of entities belonging to them.*

Despite its pervasiveness, today ontology-matching is still largely conducted by hand, in a labor-intensive and error-prone process. The manual matching has now become a key bottleneck in building large-scale information systems. Hence, the development of tools to assist in the ontology matching process has become crucial for the success of a wide variety of applications [9].

There are different algorithms for implementing the matching process, which can be generally classified along two dimensions [8]. On the one hand, there is a distinction between schema-based and instance-based matching. A schema-based matcher takes different aspects of the concepts and relations in the ontologies and uses some similarity measure to determine correspondences. An instance-based matcher takes the instances which belong to the concepts in the ontologies and compares them to discover similarity between the concepts. On the other hand, there is a distinction between element-level and structure-level matching. An element-level matcher compares properties of the particular concept or relation, such as the name, and uses them to find similarities. A structure-level matcher compares the structure of the ontologies to find similarities. These matchers can also be combined.

In Sect. 2.5, a proposal for an ontology-matching model that combines several of these elements is presented. In this chapter, the *ontology-matching* model is defined formally as:

- $ontology\text{-}matching : c \rightarrow O_i$,
- $ontology\text{-}matching(c) = \{O_X, O_Y\}$, where c is a concept (it could be not only one concept, but more than one), O_X and O_Y are domain ontologies related to the same field than c.

2.2.2 Software Agents

Besides ontologies, software agents will play a fundamental role in building the Semantic Web of the future [16]. When data is marked up using ontologies, software agents can better understand the semantics and therefore more intelligently locate and integrate data for a wide variety of tasks.

Several authors propose different definitions for an agent [21, 31]. According to [27], an agent is everything that senses and acts on its environment, modifying it.

A software agent can therefore be considered as an autonomous software entity that can interact with its environment.

There are different types of agents such as [18]:

- Autonomous agent: it has the ability to decide when action is appropriate without the need of human intervention.
- Cooperative agent: it can interact with other agents or humans via a communication language.
- Mobile agent: it is able to migrate from one computer to another autonomously and continue its execution on the destination computer.
- Intelligent agent: it has the ability to learn the user preference and adapt to external environment.

A new emergent category, intelligent web (or personal) agents, rather than doing everything for a user, would find possible ways to meet user needs and offer the user choices for their achievement. A web agent could offer several possible ways to get what a user needs on the Web. A personal software agent on the Semantic Web must be capable of receiving some tasks and preferences from a user, seeking for information from web sources, communicating with other agents, comparing information about user requirements and preferences, selecting certain choices, and finally providing answers to the user [3].

An important distinction is that agents on the Semantic Web will not act in a completely autonomous way, but rather they will take care of the heavy load in the name of their users. They will be responsible for conducting the investigation, with the obligation to present the results to the user, so that he or she can make his or her decisions.

2.2.3 Artificial Neural Networks

There is no universally accepted definition of what Artificial Neural Networks (ANNs) are, or what they should be. They can be loosely defined as large sets of interconnected simple units which execute in parallel to perform a common global task. These units usually undergo a learning process which automatically updates network parameters in response to a possibly evolving input environment.

ANNs are information processing systems inspired by the ability of the human brain to learn from observations and to generalize by abstraction [14], according to the following characteristics:

- Knowledge is acquired by the network through a learning process.
- Connection strengths between neurons, known as synaptic weights, are used to store the knowledge.

Neural models have certain common characteristics. They are given a set of inputs $x = (x_1, \ldots, x_n) \in \Re^n$ and their corresponding set of target outputs $t = (t_1, \ldots, t_m) \in \Re^m$ for a certain process. The assumption of an ANN model is that

the process that produces the output response is given by some unknown mathematical relationship $t = G(x)$ for some unknown, generally nonlinear, function G. Therefore, a candidate activation function AF (for G) is chosen and the approximation is performed using a given set of examples, named patterns; that is to say, a pattern consists of some inputs x and their associated target outputs t. The patterns are used to feed the ANN model, which contains a set of processing elements (called neurons) and connections between them (called synaptic weights). Each neuron has an activation function AF, which process the incoming information from other neurons.

Neural models may be considered as a particular choice of classes of functions $AF(x, w)$ where w are the parameters and specific procedures for training the network [25]. Training is similar to an optimization process where internal parameters of the neural model are adjusted, to fit the training data.

Training a neural network means adapting its connections so that the model exhibits the desired computational behavior for all input patterns. The process usually involves modifying the weights. Selection of training data plays a vital role in the performance of a supervised ANN. The number of training examples used to train an ANN is sometimes critical to the success of the training process.

If the number of training examples is not sufficient, then the network cannot correctly learn the actual input–output relation of the system. If the number of training examples is too large, then the network training time could be too long. For some applications, training time is a critical variable. For others, the training can be performed off-line and more training data are preferred over using insufficient training data to achieve greater network accuracy. Generally, rather than focusing on volume, it is better to concentrate on the quality and representational nature of the data set. A good training set should contain routine, unusual and boundary-condition cases [14].

2.3 ANNs and the Semantic Web: Literature Review

In this section, a review of current related work is presented, divided into two main research areas: searching and query answering on the Web, and proposals for ontology-matching using neural network models.

2.3.1 Searching and Query Answering on the Semantic Web

Considering the existence of webpage metadata, the problem of dynamic knowledge discovery in open distributed contexts has been addressed in the Helios Project [4]. Examples of open distributed contexts are Semantic Grids and Peer-based systems, where a set of independent peer nodes without prior reciprocal knowledge and no degree of relationship, dynamically need to cooperate by sharing their resources (such as data, documents or services). In the Helios Project, the authors assume that

no centralized authority manages a comprehensive view of the resources shared by all the nodes in the system, due to the dynamics of the collaborations and variability of the requirements.

On the web, information is not described by a global schema and users used to query the web using their own terminology. Then, a semantic query answering system on the web has to rewrite the query with respect to available ontologies in order to use reasoning for providing answers. An example of this system is PowerAqua [20]. This system was developed to exploit the availability of distributed, ontology-based semantic markup on the web to answer questions posed in natural language. It does not assume that the user has any prior information about the semantic resources.

The problem of answering queries considering metadata has been studied also in Peer-based systems, mainly addressing the problem of routing queries over a network. For example, Edutella [23] provides an infrastructure for sharing metadata in RDF format. The network is segmented into thematic clusters. In each cluster, a mediator semantically integrates source data. The mediator handles a request either directly or indirectly: directly, by answering queries using its own integrated schema; indirectly, by querying other cluster mediators by means of a dialog-based query processing module.

2.3.2 Ontology-Matching and ANN Models

In response to the challenge of ontology-matching on the Semantic Web and in numerous other application contexts, several proposals have appeared lately, which apply machine learning techniques to create semantic mappings.

In a recent ontology matching state-of-the-art review [11], some tools based on ANNs are addressed that using information regarding ontology schemas and instances, produce rules for ontology integration in heterogeneous databases.

Several types of ANNs have been used for various tasks in ontology matching, such as discovering correspondences among attributes via categorization and classification, or learning matching parameters, such as matching weights, to tune matching systems with respect to a particular matching task [12].

Given schema-level and instance-level information, it is sometimes useful to cluster this input into categories in order to lower the computational complexity of further manipulations with data. The self-organizing map network can be used for this purpose where the neurons in the network are organizing themselves according to the characteristics of given input patterns. This kind of neural model is used in the X-SOM ontology mapper [7] which uses a neural network model to weight existing matching techniques, combined with reasoning and local heuristics aimed to both discover and solve semantic inconsistencies.

ANNs have been used to automatically recognize the connection between web pages with similar content and to improve semantic information retrieval together with synonyms thesaurus [33], or based on text documents [30]. SEMantic INTegrator (SEMINT) is a tool based on neural networks to assist in identifying attribute

correspondences in heterogeneous databases [19]. It supports access to a variety of database systems and utilizes both schema-level and instance-level information to produce rules for matching corresponding attributes automatically.

During matching, different semantic aspects such as concept names, concept properties, and concept relationships, contribute in different degrees to the matching result. Therefore, a vector of weights has to be assigned to these aspects, which is not a trivial task and current research work depends on human heuristics. In [17], an ANN model learns and adjusts those weights, with the purpose of avoiding some of the disadvantages in both rule-based and learning-based ontology matching approaches, which ignore the information that instance data may provide. Similar to this idea, the work of [5] uses instances to learn similarity between ontology concepts to create then a newly combined ontology.

In the GLUE system [9], learning techniques are used to semi-automatically create semantic mappings between ontologies, finding correspondences among the taxonomies of two given ontologies: for each concept node in one taxonomy, the GLUE system uses a machine-learning classifier to find the most similar concept node in the other taxonomy.

Differently from the previous cited works, in [28] it is considered that labels at ontologies are human identifiers (names) for instances, normally shared by a community of humans speaking a common language inside a specific field of knowledge. It can be inferred that, if labels are the same, the instances associated to them are probably also the same or are semantically related [10]. A concept, on the other hand, can also be defined as representative of a set of instances. We can, therefore, infer that concepts that have the same instances are the same. And vice versa, instances that have the same mother concept (label) are similar. The assumption is that, to perform its tasks efficiently, an agent should be specialized in a specific domain context where it is assumed that different ontologies will manage similar concepts and content.

Those facts, combined with supervised learning by example (i.e. ANNs), can be better used by an agent to provide a more effective and more efficient response, compared to traditional retrieval mechanisms. This way, a mobile agent responsible for searching the web would not loose time and would only visit domains that could provide an answer and total response time would diminish because the knowledge would be available just-in-time for the system users. A proposal for the ontology-matching task using an ANN-based model is presented in detail in Sect. 2.5.

2.4 Web Knowledge Source Discovery

The simplest knowledge discovery mechanism is based on the traditional query/answer paradigm, where each part acts as both client and server, interacting with other nodes directly sending queries or requests, and waiting until receiving an answer. This is only possible if the domains are previously known to each other or if a collaboration relationship has already been established between them. When this

is not the case, the discovery of knowledge is affected by the dynamism of the system. Some nodes join and some nodes leave the network, at any time. Besides, each domain is responsible for its own knowledge representation and management, because there are no a-priori agreements regarding ontology language nor granularity. A typical example of such a system is the Semantic Web [3].

In open distributed systems, several nodes (domains), probably distributed among different organizations, need resources and information (i.e. data, documents, services) provided by other domains in the net. Therefore, an open distributed system can be defined as networks of several independent nodes, having different roles and capacities. In this scenario, a key problem is the dynamic discovery of knowledge sources, understood as the capacity of finding knowledge sources in the system about resources and information that, in a given moment, better response the requirements of a node request [4].

When data is marked up using ontologies, software agents can better understand the semantics and therefore more intelligently locate and integrate knowledge for a wide variety of tasks. The following examples show two motivating scenarios for knowledge discovery on the Semantic Web:

- Scenario 1 (adapted from [9]):

  Suppose a researcher wants to find out more about someone met at a conference, whose last name is Cook and teaches Computer Science at a nearby (unknown) university. It is also known that he just moved to the US from Australia, where he had been an associate professor. On the web of today, it will be difficult finding this person, because the above information is not contained within a single webpage, thus making keyword search ineffective. On the Semantic Web, however, one should be able to quickly find the answers. A marked-up directory service makes it easy for a personal software agent to find nearby Computer Science departments. These departments have marked up data using some ontology, that includes courses, people, and professors. Professors have attributes such as name, degree, and institution. Such marked-up data makes it easy for the agent to find a professor with the last name Cook. Then by examining the attribute institution, the agent quickly finds the CS department in Australia. Here, the agent learns that the data has been marked up using an ontology specific to Australian universities and that there are many entities named Cook. However, knowing that associate professor is equivalent to senior lecturer, it can select the right subtree in the departmental taxonomy, and zoom in on the old homepage of the conference acquaintance.

- Scenario 2 (adapted from [28]):

  A researcher from an European university wants to identify potential partners in some American universities to collaborate on a project answering from the calling of the European Union Framework Program.[1] *The main topic for the project is the Semantic Web, which is an unknown topic for the researcher and*

[1] http://cordis.europa.eu/fp7.

therefore he knows no colleague who can help in the research. In addition, due to the project requirements, a senior researcher has to be contacted. Similarly to the previous scenario, this problem could be more easily solved in the Semantic Web by an agent capable of dynamically discovering the appropriate sources of knowledge, by dynamically matching ontologies belonging to different universities in Europe and America, looking for researchers in the Semantic Web topic.

Searching on the Semantic Web differs in several aspects from a traditional web search, specially because of the structure of an online collection of documents in the Semantic Web, which consists of much more than HTML pages. The semantics associated to the languages for the Semantic Web allows the generation of new facts from existing facts, while traditional databases just enumerate all available facts.

Traditional search machines do not try to understand the semantics of the indexed documents. Conventional retrieval models, in which the retrievals are based on the matching of terms between documents and the user queries, is often suffering from either missing relevant documents which are not indexed by the keywords used in a query, but by synonyms; or retrieving irrelevant documents which are indexed by unintended sense of the keywords in the query.

Instead, search agents for the Semantic Web should not only find the right information in a precise way, but also should be able to infer knowledge and to interact with the target domain to accomplish its duty [24]. In the next subsection, a proposal for a knowledge source discovery agent is presented.

2.4.1 A Knowledge Source Discovery Agent

In [28], an architecture for discovering knowledge sources on the Semantic Web was proposed. This architecture is shown in Fig. 2.1. The main components of the architecture are: the mobile agents, the Knowledge Source Discovery (KSD) agent, and the domains.

The mobile agents receive the request from the user and look for an answer visiting the domains according to a list of possible domains generated by the KDS. An example of a request for the previously described Scenario 2 could be: *Which researchers from Latin American universities work on the field of ontologies and neural networks?*

The KSD agent has the responsibility of knowing which domains can provide knowledge inside a specific thematic cluster [23] and to indicate a route to mobile agents on the web carrying a user request. The KSD agent just knows the location (i.e. the url) of domains that can provide knowledge, but it does not provide the knowledge, nor the analysis what the domain contains (i.e. files, pictures, documents, etc.).

The KSD agent is specialized in a determined field of knowledge. It has to be aware of all the domains related to this field. To do that, it periodically sends

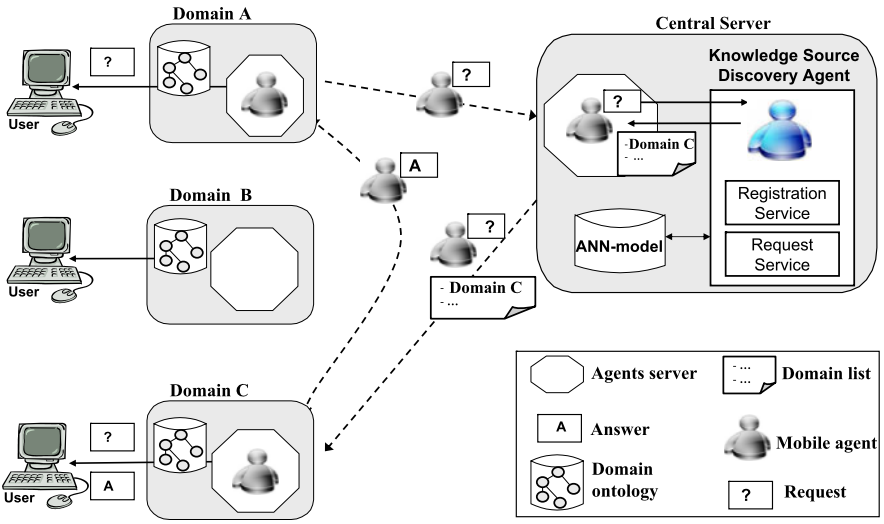

Fig. 2.1 A reference architecture for discovering knowledge sources [28]

crawlers to index websites (Google-search engine style), or the domains can register with the KSD agent when they want to be reachable (Yahoo-directory style). This task is carried out by the *Registration Service*.

Finally, the last components of the architecture are the domains. Each domain has its own ontology used to semantically markup the information published in its websites. An example of the domain content is shown in Fig. 2.2. Let us suppose that there are three domains (A, B, and C) which belong to the Research & Development (R + D) field of knowledge. The domain A uses the KA-ontology[2] provided by the free open source ontology editor Protege. The domain B uses the SWRC-ontology,[3] Semantic Web for Research Communities, which is an ontology for modeling entities of research communities and their relationships. Finally, the domain C uses an own-highly-specialized ontology model. All of these ontologies are implemented in Web Ontology Language (OWL).

In addition, RDF is used to define an ontology-based semantic markup for the domain website. Each RDF-triple assigns entities and relations in the text linked to their semantic descriptions in an ontology. For example, in the domain A, the following RDF-triples: (*O.C., interest, Semantic Grid*), (*O.C., interest, Semantic Web*) and (*O.C., interest, Ontological Engineering*) represent the research interests of O.C. described in the text (see the bottom area of Fig. 2.2). As can be seen in the figure, each domain may use a different ontology to semantically annotate the provided information even if they belong to the same field of knowledge.

The KSD agent is capable of identifying dynamically which domains could satisfy a request brought to it by a mobile agent. This dynamic knowledge discovery

[2] http://protege.cim3.net/file/pub/ontologies/ka/ka.owl.

[3] http://ontoware.org/projects/swrc/.

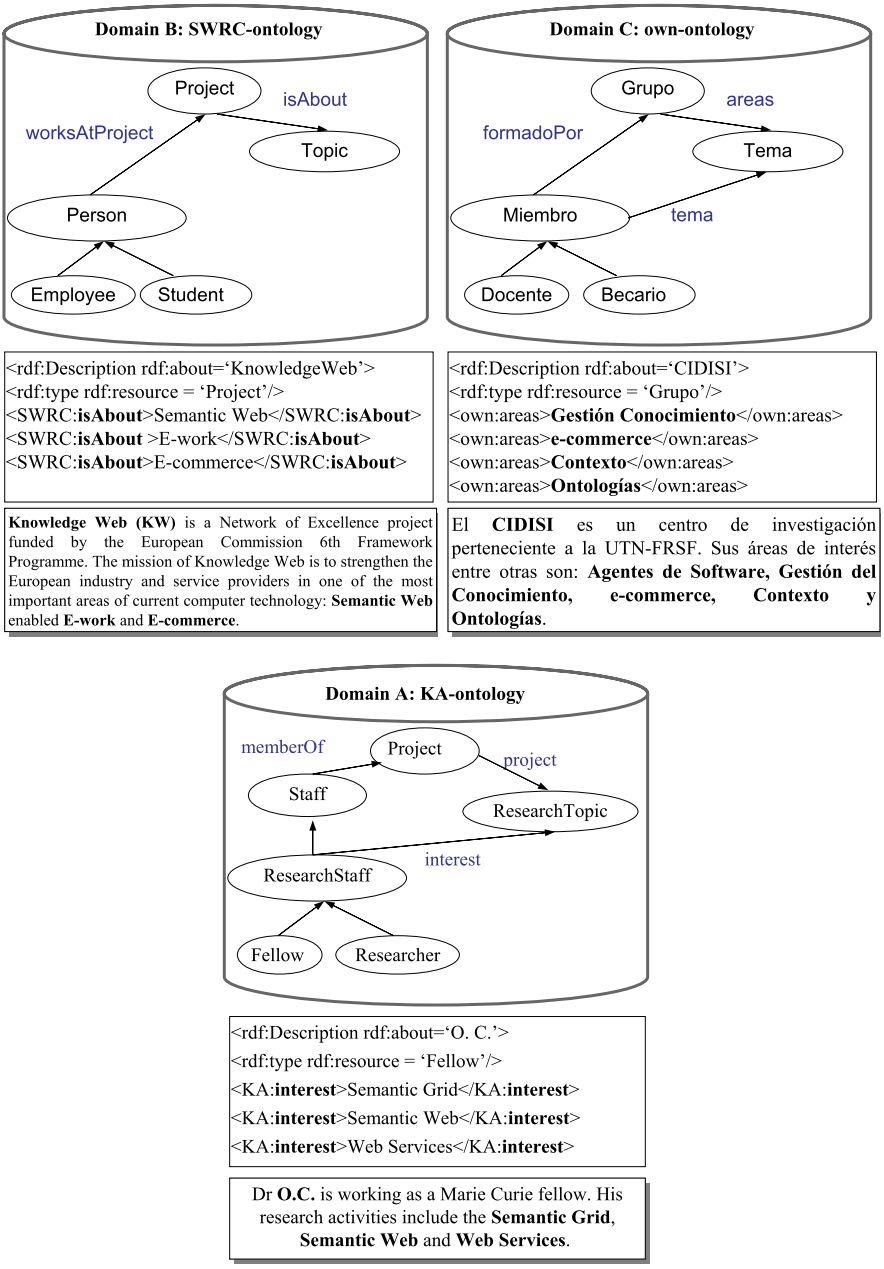

Fig. 2.2 Domains belonging to the R + D field and their corresponding annotated ontologies

requires models and techniques that allow to find ontology concepts that have semantic affinity among them, even when they are different syntactically. In order to do its responsibility efficiently, the KSD agent has to be able to match (probably different) domain ontologies. To face this ontology-matching problem, we propose the use of a machine learning approach, in particular an ANN model with supervised learning which is trained (and re-trained periodically) off-line with the data retrieved as previously stated. The ANN-based matching model is stored in the KSD agent Knowledge Base (KB) and it is explained in detail in the next section.

2.5 The ANN-Based Ontology-Matching Model Inside the KSD Agent

This section of the chapter presents an ANN-based model that can be used for ontology-matching purposes in the Semantic Web.

An ANN model can be classified according to the type of connections among their neurons. A network is named feedforward if the flow of data is from inputs to outputs, without feedback. Generally this topology has distinct layers such as input, hidden and output, with no connections among neurons belonging to the same layer. Inside the feedforward models, which are the most widely used, there is a model named multi-layer perceptron (MLP).

The MLP model consists of a finite number of units called perceptrons (Fig. 2.3), where each unit of each layer is connected to each unit of the subsequent/previous layer. These connections are called links or synapses and they only exist between layers. The signal flow is unidirectional, from the inputs to the outputs, from one

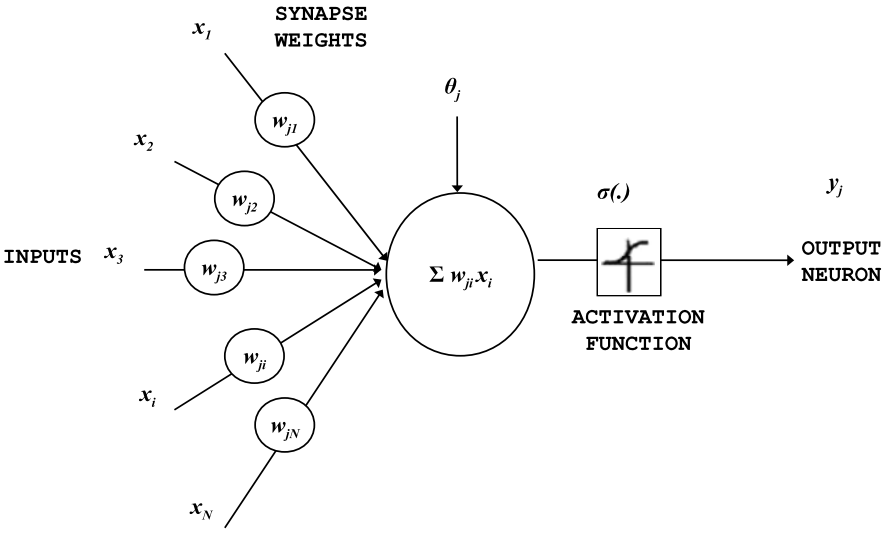

Fig. 2.3 MLP neuron model

layer to the next one, thus the term feedforward. The rules for the MLP model are the following:

- The jth neuron of the kth layer receives as input each x_i of the previous layer. Each value x_i is then multiplied by a corresponding constant, called weight w_{ji}, and then all the values are summed.
- A shift θ_j (called threshold or bias) is applied to the above sum, and over the results an activation function σ is applied, resulting in the output of the jth neuron of the kth layer. The MLP neuron model is the following:

$$y_j = \sigma\left(\sum_{i=1}^{N} w_{ji} x_i + \theta_j\right) \tag{1}$$

where $i = 1, \ldots, N$, $j = 1, \ldots, M$, and some common choices for σ can be one of the following: the logistic sigmoid (equation (2)), the hyperbolic tangent (equation (3)) or the linear function (equation (4)).

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{3}$$

$$y(x) = x. \tag{4}$$

The number of layers and neurons in each layer are chosen a-priori, as well as the type of activation functions for the neurons. The number of neurons in the hidden layer is usually determined by trial-and-error in order to find the simplest network that gives acceptable performance. Then, the values of the weights w_{ji} and bias θ_j are initialized. These values are chosen so that the model behaves well on some set (named training set) of inputs and corresponding outputs. The process of determining the weights and thresholds is called learning or training.

In the MLP model, the learning is supervised and the basic learning algorithm used is called backpropagation [26, 29] which uses gradient descend to minimize the cost function (the error E) of the ANN model, generally defined as the mean square error (mse) between the desired output (targets) and the actual network output. During learning, the mse propagates backwards through the network, hence the term backpropagation, across the layers of the network model and the weights are changed accordingly to the error mismatch between the actual network outputs y and the target outputs t over all the training patterns [14].

A major result regarding MLP models is the so-called *universal approximation theorem* [6] which states that given enough neurons in the hidden layer, an MLP neural model can approximate any continuous bounded function to any specified accuracy; in other words, there always exists a three-layer MLP neural network which can approximate any arbitrary nonlinear continuous multidimensional function to any desired accuracy, provided that the model has enough neurons [15]. That is why this model has been more extensively studied and used during last years. It is worth noting, however, that the theorem does not say that a single-layer network is

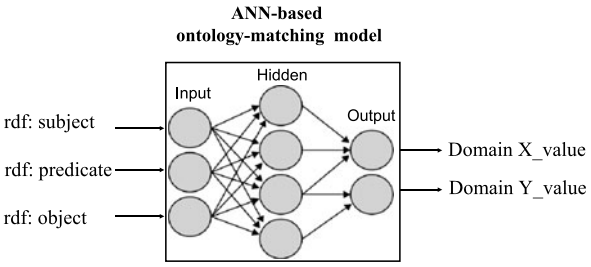

Fig. 2.4 ANN-based ontology-matching model for a KSD agent

optimum in the sense of learning time or ease of implementation; moreover, the theorem does not give indications about the required number of hidden units necessary in order to achieve the desired degree of accuracy.

For neural networks, a matching problem can be viewed as a classification problem. The input to our problem includes RDF-triples instances belonging to the RDF annotations of the A KA-ontology domain, and similarly for domain B and C. We address this problem using machine learning techniques as follows: our ANN-based matcher uses schema-level information and instance-level information inside the A ontology to learn a classifier for A, and then it uses schema-level information and instance-level information inside the B ontology to learn a classifier for B. It then classifies instances of B according to the A classifier, and vice-versa. Hence, we have a method for identifying instances of $A \cap B$. The same idea is applied for more than two domains. Our approach just indicates which domains may be able to respond to a query, but it does not provide a similarity measurement for the analyzed ontology terms, differently from [10] where an ANN model is used for providing or combining similarity measures among heterogeneous ontology terms.

Applying machine learning to this context raises the question of which learning algorithm to use and which types of information to use in the learning process. Our model (a schematics representation of the general ANN-based ontology-matching model can be seen in Fig. 2.4) uses the standard backpropagation algorithm for supervised learning, which means that, for each input data point presented to the ANN model, there must be a corresponding matching output or target to be related with [32]. These {input/output} pairs are named training patterns.

Given a number of ontologies and their instances (belonging to the same field of knowledge), i.e. ontology X and ontology Y, training patterns are formed to train the proposed ANN-model by analyzing the RDF-triples belonging to the annotations of the ontology domains, as follows:

Input pattern: $\langle rdf{:}subject; rdf{:}predicate; rdf{:}object \rangle$,
Target pattern: $\langle DomainX_{value}; DomainY_{value} \rangle$

where the target pattern is a vector: the first position stands for ontology domain X and the second position stands for ontology domain Y. The value of each position in the vector indicates whether the input RDF-triple exists in the corresponding ontology domain (1) or not (0). This is better explained with a example in the next subsection.

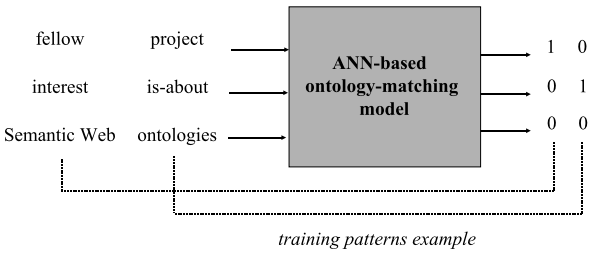

Fig. 2.5 ANN-based ontology-matching model: training patterns example

2.5.1 ANN-Based Ontology-Matching Model: Training Phase

Using the available data from all the domains (label names and associated instances from the RDF-triples used as semantic markup of the websites, as shown in Fig. 2.2), the ANN-based ontology matching model training patterns are formed.

Each output neuron is specialized in recognizing RDF-triples belonging to the domain ontology that the neuron represents. The label and instance strings are codified using the standard ASCII code and are then normalized into the activation function domain of the hidden neurons, before entering the model, because this significantly improves training time and model accuracy. A simple example of two training patterns is presented in Fig. 2.5.

For example, considering all three ontologies presented in Fig. 2.2, a training pattern indicating that the RDF-triple $\langle$*fellow*; *interest*; *SemanticWeb*$\rangle$ can be found on the Domain A ontology but not on B nor C would be:

Input pattern: $\langle$*fellow*; *interest*; *SemanticWeb*$\rangle$,
Target pattern: $\langle 1; 0; 0\rangle$.

This means that given the fact that there are fellows in the domain A whose research interest is the Semantic Web, its corresponding RDF-triple would be $\langle$*fellow*; *interest*; *SemanticWeb*$\rangle$ and its corresponding output target would be $\langle 1; 0; 0\rangle$: only the first vector value (that represents Domain A is equal to 1) indicating that this triple can be found on domain A ontology. The second training pattern indicates that the RDF-triple $\langle$*project*; *is-about*; *ontologies*$\rangle$ can be found on domain B ($\langle 0; 1; 0\rangle$), because the second value of the target vector is equal to 1.

The MLP model parameters are set according to typical values, randomly initialized. The number of input neurons for the MLP model is set to a standard RDF-triple. The hidden layer neurons number is set empirically, according to the training data and a desired accuracy for the matching. At the output, there is a specialized output neuron in the model for each domain, that recognizes when a domain ontology label or instance is presented to the model. The allowed values for each output neuron are 1 or 0, meaning that the neuron recognizes/does not recognizes a concept belonging to the domain it represents.

The good generalization property of an ANN model means the ability of a trained network to correctly predict a response to a set of inputs not seen before. It says that a trained network must perform well on a new dataset distinct from the one used for training.

The ANN-based ontology-matching model here proposed is trained with each domain ontology RDF-annotations and their corresponding instances. Our basic assumption is that knowledge is captured in an arbitrary ontology encoding, based on a consistent semantics. From this, it is possible to derive additional knowledge such as, in our case, similarity of concepts in different ontologies. Understanding that labels describe concepts in natural language one can derive that concepts/instances having the same labels are similar. This is not a rule which always holds true, but it is a strong indicator for similarity. Other constructs as subclass relations or type definition can be interpreted similarly.

We have used a small set of patterns (34) for ANN model training, because at the moment we are working with small domains and with not highly populated ontologies. For testing model effectiveness, several ANN models have been tested, considering different number of hidden neurons and performance on training and validation patterns, using the cross-validation procedure combined with the Levenberg–Marquardt [22] training algorithm, which is a quasi-Newton method, designed to approach second-order training speed without having to compute a Hessian matrix. When the ANN model performance function has the form of a sum of squares (typical in feedforward networks training), then the Hessian matrix can be approximated as $H = J^T J$ and the gradient can be computed as $g = J^T e$, where J is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix [14]. The ANN model is built and trained off-line, and its parameters are tuned and re-trained periodically when data changes or new domains are added to the system. However, the time needed for this processes can be disregarded because the ANN model is used on-line. Examples of model use for matching are presented in the next subsection.

2.5.2 ANN-Based Ontology-Matching Model: Matching Phase

The Knowledge Source Discovery agent uses the ANN model on-line once the model has been trained, producing and instant response, each time a mobile agent carrying a request knocks on its door. The query is expressed in natural language by an end-user, and it is processed at the mobile agent, until only the keywords remain. These keywords, represented as a RDF-triple, are used to consult on the ANN model, as shown in the example of Fig. 2.6. In the presented example, the ANN model indicates that the domain ontologies of A and B contain some ontology labels or instances that are similar to the presented request. Therefore, the Knowledge Source Discovery agent should return to the agent a Domain-list containing the domains A and B to be visited by the mobile agent. According to the ANN model of the KSD agent, those domains are very likely to be able to provide the required information.

Because of how neural models work (interpolating data) and the properties associated to a three-layer MLP model (generalization ability), the model would always

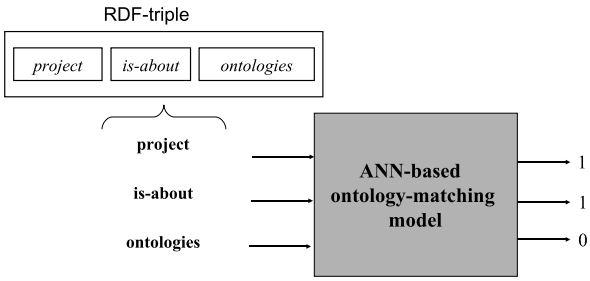

Fig. 2.6 Querying the ANN-based ontology-matching model

Table 2.1 Query RDF-triplets examples

Query RDF-triplets	Matching domain ontologies RDF-triplets
(1) ⟨*fellow*, *interest*, *semanticWeb*⟩	Domain A: ⟨*fellow*, *interest*, *semanticWeb*⟩
(2) ⟨*researcher*, *topic*, –⟩	Domain B: ⟨*researcher*, *topic*, *semanticWeb*⟩
(3) ⟨*miembro*, *tema*, *gobierno*⟩	Domain C: ⟨*miembro*, *tema*, *gobiernoElectronico*⟩
(4) ⟨*project*, *is-about*, *ontologies*⟩	Domain A: ⟨*fellow*, *interest*, *ontologies*⟩,
	Domain B: ⟨*researchProject*, *is-about*, *ontologies*⟩
(5) ⟨*researcher*, *topic*, *web*⟩	Domain B: ⟨*researcher*, *topic*, *semanticWeb*⟩
(6) ⟨–, –, *semanticGrid*⟩	Domain A: ⟨*fellow*, *interest*, *semanticGrid*⟩

provide a response [14]. Besides, standard learning algorithms can be used for the model, for which stability and convergence is guaranteed. If the request contains concepts totally unknown to the model, it shows that fact answering with values very different from the training patterns.

The ANN-based ontology-matching model has been tested on six query-examples, presented in Table 2.1. Note that the query triplet (4) ⟨*project*, *is-about*, *ontologies*⟩ has a translation in both domain A and domain B ontologies, therefore the ANN model should indicate that the domain ontologies of *A* and *B* contain some ontology labels or instances that are similar to the presented request. Therefore, the KSD agent should return a Domain-list containing the domains *A* and *B* to be visited by the mobile agent carrying the request, because according to the ANN model of the KSD agent, those domains are very likely to be able to provide the required information. Another interesting test query is represented by (3) ⟨*miembro*, *tema*, *gobierno*⟩ and (5) ⟨*researcher*, *topic*, *web*⟩, where the matching model must recognize an instance name which is part of an ontology instance name. For all of these examples the ANN-based ontology matching model has provided a satisfactory result.

The number of query triplets is fixed a priori for each query category, however the final number of ontology triplets for training the ANN-model is dependent on how populated the ontologies are. Therefore, triplets must be created off-line for training the model, but it then can be used on-line for query resolution purposes. Linguistic terms can be mapped into ontology classes (i.e., researcher) or instances (i.e., semanticWeb, agents).

2.6 Conclusions

This chapter has presented some basic concepts and foundations regarding the new Semantic Web, how it will be populated with ontologies and why ontology-matching techniques are needed. The idea of software agents that travel the web carrying query request from users has also been addressed. The web knowledge source discovery task has been explained in detail and some motivating scenarios were introduced.

A condensed literature review on these subjects has been presented, and the advantages of combination of Artificial Intelligence techniques and models, such as agent-based technologies and Artificial Neural Networks (ANN), that can be used for intelligent web knowledge source discovery in the emergent Semantic Web, have been highlighted.

In summary, this chapter has presented the benefits of combining Artificial Neural Networks with Semantic Web Technologies, and an ANN-based software agent with capabilities for discovering distributed knowledge sources has been presented.

References

1. Baeza-Yates, R.: Web mining. In: Proc. LA-WEB Congress, p. 2 (2005)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **5**(1), 29–37 (2001)
3. Breitman, K., Casanova, M.A., Truszkowski, W.: Semantic Web: Concepts, Technologies and Applications. Springer, London (2007)
4. Castano, S., Ferrara, A., Montanelli, S.: Dynamic knowledge discovery in open, distributed and multi-ontology systems: techniques and applications. In: Web Semantics and Ontology. Idea Group Inc, London (2006)
5. Chortaras, A., Stamou, G.B., Stafylopatis, A.: Learning ontology alignments using recursive neural networks. In: Proc. Int. Conf. on Neural Networks (ICANN), Poland. Lecture Notes in Computer Science, vol. 3697, pp. 811–816. Springer, Berlin (2005)
6. Cybenko, G.: Neural networks in computational science and engineering. IEEE Computational Science and Engineering **3**(1), 36–42 (1996)
7. Curino, C., Orsi, G., Tanca, L.: X-SOM: Ontology mapping and inconsistency resolution. In: 4th European Semantic Web Conference (ESWC'07), 3–7, June 2007
8. Davies, J., Studer, R., Warren, P.: Semantic Web Technologies: Trends and Research in Ontology-Based Systems. Wiley, London (2007)
9. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Ontology matching: A machine learning approach. In: Handbook on Ontologies in Information Systems, pp. 385–403. Springer, New York (2004)
10. Ehrig, M., Sure, Y.: Ontology mapping—an integrated approach. In: Proc. 1st European Semantic Web Symposium (ESWS 2004), Greece. Lecture Notes in Computer Science, vol. 3053, pp. 76–91. Springer, Berlin (2004)
11. Euzenat, J., Barrasa, J., Bouquet, P., Bo, J.D., et al.: State of the art on ontology alignment. D2.2.3, Technical Report IST-2004-507482, KnowledgeWeb, 2004
12. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, London (2007)
13. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering—with Examples from the Areas of Knowledge Management, e-Commerce and the Semantic Web. Springer, London (2004)

14. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall, New York (1999)
15. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2**(5), 359–366 (1989)
16. Hendler, J.: Agents and the Semantic Web. IEEE Intelligent Systems **16**(2), 30–37 (2001)
17. Huang, J., Dang, J., Vidal, J., Huhns, M.: Ontology matching using an artificial neural network to learn weights. In: Proc. IJCAI Workshop on Semantic Web for Collaborative Knowledge Acquisition (SWeCKa-07), India (2007)
18. Lam, T., Lee, R.: iJADE FreeWalker—an intelligent ontology agent-based tourist guiding system. Studies in Computational Intelligence **72**, 103–125 (2007)
19. Li, W., Clifton, C.: SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. Data and Knowledge Engineering **33**(1), 49–84 (2000)
20. López, V., Motta, E., Uren, V.: PowerAqua: fishing the semantic web. In: Proc. 3rd European Semantic Web Conference, Montenegro. Lecture Notes in Computer Science, vol. 4011, pp. 393–410. Springer, Berlin (2006)
21. Maes, P.: Intelligent software. Scientific American **273**(3), 84–86 (1995)
22. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. SIAM Journal on Applied Mathematics **11**, 431–441 (1963)
23. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T.: EDUTELLA, A P2P networking infrastructure based on RDF. In: Proc. 11th World Wide Web Conference (WWW2002), USA, pp. 604–615 (2002)
24. Peis, E., Herrera-Viedma, E., Montero, Y.H., Herrera, J.C.: Ontologías, metadatos y agentes: Recuperación semántica de la información. In: Proc. II Jornadas de Tratamiento y Recuperación de la Información, España, pp. 157–165 (2003)
25. Pinkus, A.: Approximation theory of the MLP model in neural networks. Acta Numerica **1**, 143–195 (1999)
26. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**, 533–536 (1986)
27. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, New York (2002)
28. Stegmayer, G., Caliusco, M.L., Chiotti, O., Galli, M.R.: ANN-agent for distributed knowledge source discovery. In: On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops. Lecture Notes in Computer Science, vol. 4805, pp. 467–476. Springer, Berlin (2007)
29. Werbos, P.: The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting. Wiley, New York (1994)
30. Wermter, S.: Neural network agents for learning semantic text classification. Information Retrieval **3**(2), 87–103 (2000)
31. Wooldridge, M.: An Introduction to Multiagent Systems. Wiley, New York (2002)
32. Wray, J., Green, G.: Neural networks, approximation theory and precision computation. Neural Networks **8**(1), 31–37 (1995)
33. Zhu, X., Huang, S., Yu, Y.: Recognizing the relations between Web pages using artificial neural network. In: Proc. ACM Symposium on Applied Computing, USA, pp. 1217–1221 (2003)

http://www.springer.com/978-1-84996-076-2

Emergent Web Intelligence: Advanced Semantic Technologies

Badr, Y.; Chbeir, R.; Abraham, A.; Hassanien, A.-E. (Eds.)

2010, XVI, 544 p., Hardcover
ISBN: 978-1-84996-076-2