ISBN 978-83-60810-22-4

A Neural Model for Ontology Matching

Emil Şt. Chifu and Ioan Alfred Leția
Department of Computer Science, Technical University of Cluj-Napoca, Barițiu 28, RO-400027 Cluj-Napoca, Romania
Email: Emil.Chifu@cs.utcluj.ro, letia@cs-gw.utcluj.ro

Abstract—**Ontology matching is a key issue in the Semantic Web. The paper describes an unsupervised neural model for matching pairs of ontologies. The result of matching two ontologies is a class alignment, where each concept in one ontology is put into correspondence with a semantically related concept in the other one. The framework is based on a model of hierarchical self-organizing maps. Every concept of the two ontologies that are matched is encoded in a bag-of-words style, by counting the words that occur in their OWL concept definition. We evaluated this ontology matching model with the OAEI benchmark data set for the bibliography domain. For our experiments we chose pairs of ontologies from the dataset as candidates for matching.**

I. Introduction

THE USAGE of software services raises the problem of discovering the relevant ones for a given purpose, and still manual effort is needed to find and compose services. To solve this problem, the researchers in Semantic Web propose to make the service descriptions more meaningful by annotating them with a semantic description of their functionality. And the meaning of these semantic descriptions is specified in a domain ontology [12].

The semantic heterogeneity problem is encountered classically in the information integration area as well as in the new domain of Semantic Web. Ontology matching allows the knowledge and data expressed in different ontologies to interoperate. To name only two situations, the interoperability is important when two agents communicate, as well as for merging two ontologies into a result, combined and still consistent single ontology. As a consequence, ontology matching is a key issue in the Semantic Web [5].

In this paper we propose an unsupervised neural model for matching pairs of ontologies. The result of matching two ontologies is a class alignment: each concept in one ontology is put into correspondence with the most related concept in the other ontology from the semantic point of view. We disregard here the problem of also matching the properties and the instances of the ontology concepts.

In order to establish a mapping of the concepts of one ontology to the concepts of the other ontology, we classify the concepts of the first ontology against the taxonomic structure of the second ontology. The classification starts from a representation of the concepts built as a result of analyzing their OWL concept definitions. We collect the words used in the semistructured textual descriptions extant in OWL ontology definition files. Based on this text mining process, we represent the ontology nodes (the concepts) as bag-of-words vectors. These vectors are used as input data for an unsupervised neural network. More specifically, our framework is based on an unsupervised training of an extended model of hierarchical self-organizing maps.

In our approach, we consider the OWL semistructured textual definition of each concept as a small text document. As such, we represent the ontology classes as text documents, like in a document categorization setting. We cast the concept classification problem as a text document classification in a vector space. The semantic content features of a concept are the frequencies of occurrence of different words in the document representation of the concept. The classification of the concepts of the first ontology into the taxonomy of the second ontology proceeds by associating every ontology 1 concept to one node of the taxonomy of ontology 2, based on a similarity in the vector space.

The taxonomy of the second ontology is given as the initial state of the neural network. The training of the unsupervised neural network takes place by exposing the initialized hierarchical self-organizing map to the vector representations of the document-like concepts of the first ontology, as extracted from their OWL definitions.

In the rest of the paper, after a review of related work, section III presents the neural network learning solution chosen and adapted in our framework. Then section IV details the architecture and implementation of our ontology matching model, and section V describes the experimental results. The conclusions and future directions are presented in section VI.

II. Related Work

A classification process algorithmically similar to our ontology matching scenario takes place in the named entity classification and taxonomy enrichment methods, where the task is to associate every named entity to one taxonomic concept or to add new concepts as subclasses attached under existent nodes (concepts) of a given initial taxonomy. From this point of view, our approach is similar to [10, 1, 13, 2]. All the methods mentioned in [10, 1, 13, 2] use a similarity based top-down classification process like in our model. The main difference from our work is that their classification is based on decision trees, whereas our classification is driven by a neural network.

The book [5] includes a comprehensive survey of ontology matching approaches and tools. Taking into account the kind of input used for computing the similarity of a concept in the first ontology to an appropriate concept in the second ontology, the authors classify the approaches as name-based (terminological) techniques, structure-based, extensional, and semantic approaches. In their turn, the terminological techniques are divided into string based and linguistic approaches.

The method described in this paper belongs to the string based approaches. In order to compute the similarity between two matched concepts we actually compute a *token-based distance* between them. The computation of token-based distances is inspired from information retrieval, and consequently a concept definition is taken as a string consisting of a multi-set of words (also called bag of words). The bag of words for each concept is a vector in a vector space, and the token based similarity is a similarity metric specific to the vector space.

This token string based category of methods is called linguistic matching in [9]. And the multi-set of words (the bag of words) is called a virtual document, which contains the definition of a concept. The approaches in this category extract the meaning of each concept from such virtual documents.

Many of the ontology matching approaches make use of external resources of common knowledge like lexicons, thesauri, and upper level or domain specific ontologies [5]. It is worth mentioning the method in [11], which dynamically select, exploit (reuse), and combine multiple and heterogeneous online ontologies in order to derive semantic mappings. The mappings are provided either by a single online ontology or by a reasoning process that uses several ontologies. By this process of harvesting the Semantic Web, the authors prove that the Semantic Web itself is a dynamic source of background knowledge that can be successfully used to solve real-world problems, including ontology matching.

III. Machine Learning Method

In our framework for ontology matching, we classify the concepts of the first ontology against the taxonomic structure of the second ontology. Our extended model of hierarchical self-organizing maps – Enrich-GHSOM – represents the unsupervised neural network based learning solution adopted by our framework. This choice is suitable to the knowledge structure onto which the concepts of the first ontology are classified – a taxonomy, i.e. an is-a hierarchy of concepts.

A. Self-organizing Maps

The Self-Organizing Map (SOM, also known as Kohonen map) learning architecture [8] is one of the most popular unsupervised neural network models. SOM can be seen as a projection method which maps a high dimensional data space into a lower dimensional one. The resulting lower dimensional output space is a rectangular SOM map, represented as a two-dimensional grid of neurons. Each input data item is mapped into one of the neurons in the map. SOM plays also the role of a clustering method, so that similar data items – represented as vectors of numerical attribute values – tend to be mapped into nearby neurons.

The SOM map learns by a self-organization process. The training proceeds with unlabeled input data like any unsupervised learning. Clusters (classes) are discovered and described by gradually detected characteristics during the training process. These gradually adjusted characteristics play the role of weights in the weight vector associated to each neuron.

B. Growing Hierarchical Self-organizing Maps

The growing hierarchical self-organizing map (GHSOM) model consists of a set of SOM maps arranged as nodes in a hierarchy and it is able to discover hierarchical clusters [4]. The SOM's in the nodes can grow horizontally during the training by inserting either one more row or one more column of neurons. This happens iteratively until the average data deviation (quantization error) over the neurons in the SOM map decreases under a specified threshold τ_1.

The SOM's in the nodes can also grow vertically during the training, by giving rise to daughter nodes. Each neuron in the SOM map could be a candidate for expansion into a daughter node SOM map (see Figure 1). The expansion takes place whenever the data deviation on the current neuron is over a threshold τ_2. The thresholds τ_1 and τ_2 control the granularity of the hierarchy learned by GHSOM in terms of depth and branching factor.

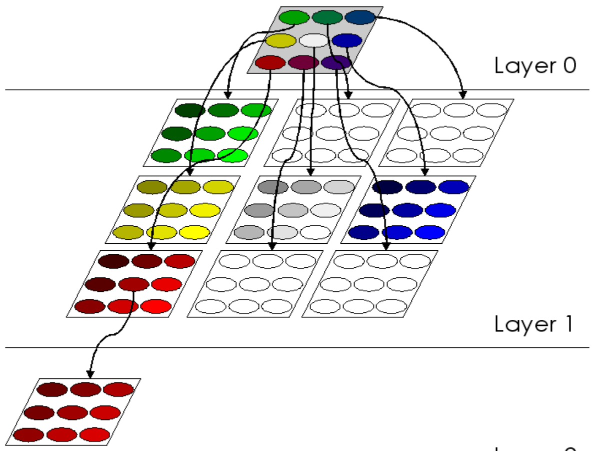

Figure 1. The GHSOM neural network model.

C. Enrich-GHSOM

The growth of a GHSOM neural network is a completely unsupervised process, being only driven by the unlabeled input data items themselves together with the two thresholds and some additional learning parameters. There is no way to suggest from outside any initial paths for the final learnt hierarchy. We have extended the GHSOM model with the possibility to force the growth of the hierarchy along with some predefined paths of a given hierarchy.

Our new extended model, Enrich-GHSOM, is doing a classification of the data items into an existing tree hierarchy structure. This initial tree plays the role of an initial state for the tree-like neural network model. The classical GHSOM model grows during the training by only starting from a single node. The top-down growth in our extended model starts from a given initial tree and inserts new nodes attached as successors to any of its intermediate and leaf nodes.

In Enrich-GHSOM, the nodes of the predefined hierarchy are labeled with some data item labels from the input data space used for training. Algorithm 1 describes the learning algorithm of the Enrich-GHSOM neural network, where mqe_j is the mean quantization error of the data items mapped on neuron j, and mqe_0 is the global quantization error of the entire training data set.

Algorithm 1: *Training the Enrich-GHSOM neural network*

```
Inputs:  predefined initial tree;
         training data space;
Output:  enriched tree;
begin
 layer $i$ = 0
 do
 {
   // training epoch associated to layer $i$:
   for all (SOM maps on layer $i$)
   {
     // phase 1:
     Train the SOM map
         // The SOM training converges
         // by satisfying threshold $\tau_1$.
     // phase 2:
     for all (neurons $j$ of the current SOM map)
     {
       if(neuron $j$ has been initialized as predefined)
       {
         Propagate the data set mapped in neuron $j$
         towards the predefined daughter map
         of neuron $j$.
       }
       else // Neuron $j$ has been initialized randomly.
       if($mqe_j > \tau_2 * mqe_0$)
       {
         Born a new daughter map from neuron $j$,
         exactly like in the GHSOM model.
       }
     }
   }
   $i$ = $i$ + 1
 }
 while( there is at least one SOM map on layer $i$ )
end
```

The training data items propagate top-down throughout the given tree hierarchy structure. When the propagation process hits a parent SOM of a tree node, then the weight vector of the corresponding parent neuron in that parent SOM is already initialized with the data item vector of that predefined daughter node label. The weight vectors of the SOM neurons with no predefined daughter are initialized with random values. Then the training of that SOM map proceeds by classifying the training data items against the initialized neurons. Training data items that are similar (as vectors) to the predefined initialized neurons are propagated downwards to the associated predefined daughter SOM nodes to continue the training (recursively) on that predefined daughter SOM. Data items that are not similar to the initialized neurons are mapped to other, non-initialized (actually randomly initialized), neurons in the same SOM. They propagate downwards only if the threshold τ_2 is surpassed, giving rise to new daughter SOM nodes.

For instance, consider the parent neuron of a current SOM node is labeled *mammal*, and there are two predefined daughter nodes labeled *feline* and *bear*, which correspond to two predefined initialized neurons in the current SOM. Then the training data item vector *dog* is not similar to any of the two neuron initializer weight vectors associated to *feline* and *bear* (see Figure 2, where the neuron initializers are marked with bold). So *dog* will remain as classified into that SOM – mapped on another, non-initialized neuron – i.e. as daughter (direct hyponym) of *mammal* and twin of the existent nodes *feline* and *bear*. Also, a data item labeled *tiger* – similar with the weight vector of the "*feline*" neuron – will be propagated into the associated predefined daughter SOM map together with other terms that correspond to felines. They will all become direct or indirect hyponyms of the concept *feline*. The process continues top-down for all the SOM nodes in the predefined initial tree hierarchy, ending at the leaves. The data item vector representations of the node labels of the given initial tree play the role of *predefined initializer weight vectors* for our neural model.

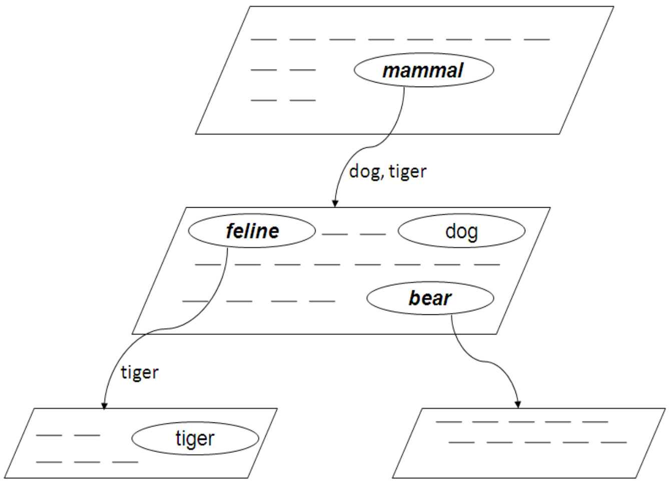

Figure 2. The Enrich-GHSOM neural network model.

For the evaluation of the classification accuracy of the Enrich-GHSOM neural network, we compare the given initial tree-like state of the network with the enriched tree-like state. The output tree is nothing else than the input tree enriched with the new nodes inserted as classified during the top-down training process. We consider a training data item as classified into, or associated to, one node of the given input tree when that SOM node is the last node of the given tree that has been traversed by the data item before leaving the tree. Obviously this is also the deepest of the input tree nodes traversed by the item during the training. After leaving the predefined input tree, the data item will only traverse

SOM nodes newly inserted during the Enrich-GHSOM training. In short, the nodes to which different data items get classified are the nodes where that data items leave the given input tree.

IV. A MODEL FOR UNSUPERVISED ONTOLOGY MATCHING

The architecture of our model is implemented as a pipeline with several processing stages. The whole processing can be divided in two main steps: the *acquisition of the vector representations of the concepts*, and then the *classification of the first ontology concepts into the taxonomy of the second ontology*.

The OWL class definition of each concept of the two ontologies being matched is considered as a small text document. For one concept, the document is the fragment broken out of the OWL ontology description, which defines only that concept. From such a semistructured document-like representation of the taxonomic node, we extract the words which compose the concept name, and also the words occurring in the name of its direct super-class. We also collect the words from the names of relations and properties for which the concept plays the role of domain, as well as the words in the concept names to which these relations point, i.e. the range concepts of the relations. Usually the words in the OWL descriptions are agglutinated, so we apply the rule of thumb that a new word begins with an uppercase letter (the camel case convention) or after an underscore or a dash.

The semantic content features of the vector representation of one concept document are the frequencies of occurrence of the different words extracted from the OWL semistructured text document representation of the concept. The words extracted in this way can be considered as a gloss definition of the concept: “An instance of the concept *Collection* is a *Book* (direct superclass of *Collection*) which has as *parts* (relation *parts*) instances of the concept *InCollection* (range concept for the relation *parts*)”. Always the gloss of a concept defines the restrictions imposed upon its direct superclass as being the ones specific to the current concept. In the example, the restriction is that the range of the property *parts* to be restricted to the concept *InCollection*. (In the ontology of the example, the property *parts* is defined as having *Part* as range concept, for which *InCollection* is only one of its subclasses.) The concepts in this example can be recognized in Figure 3, which illustrates the taxonomy of one of the ontologies actually used in our experiments reported in section V. Some of the OWL concept definitions, including the ones used in our experiments, also have a comment in natural language. This again plays the role of a gloss definition. For the example concept *Collection*, the gloss comment is “*A book that is collection of texts or articles*”.

The first ontology concepts treated as documents are mapped to the second ontology classes (concepts). The classification algorithm proceeds by “populating” the taxonomy of the second ontology with the first ontology concept documents. The *Enrich-GHSOM* neural network drives a top-down hierarchical classification of the first ontology concepts along with the taxonomy branches of the second ontology. Every ontology 1 concept is associated to one node of ontology 2.

In order to use our Enrich-GHSOM neural network to induce such a classification behavior, a symbolic-neural translation is first done by parsing a textual representation of the second ontology taxonomy, which has the form of *is_a*(*concept*, *superconcept*) assertions. The result of this parsing is the initial internal tree-like state of the neural network, which mirrors the taxonomy of the second ontology. In order for the initialized network to be able to classify ontology 1 concept documents into this (ontology 2) taxonomic structure, a representation as a numerical vector is needed for each node in this taxonomy. This node vector plays the role of predefined initial weight vector for the neural network (see section III-C). Actually, for each of the second ontology nodes, we define this vector as being the document vector representation of the concept associated to the node.

During the top-down Enrich-GHSOM classification process, vector similarity computations take place between the vector representations of the classified documents (which represent concepts of ontology 1) on one hand, and the vector representations of the documents which represent the taxonomic nodes (of ontology 2) traversed by the ontology 1 concepts on the other hand. The union of all the words used as features for the classified ontology 1 concepts and for the taxonomy 2 nodes constitutes a global vocabulary.

A. Vector Representation

Since Enrich-GHSOM is a neural network system, the ontology 1 concepts classified by Enrich-GHSOM and the concepts of taxonomy 2 have to be represented as vectors. In our framework, the features of the vector representation of a concept encode semantic content information in an $\mathbf{R}^n$ vector space. Specifically, the features are the frequencies of occurrence of different words in the document representing the concept.

In such a setting, the meaning of semantically similar concepts is expressed by similar vectors in the vector space. The Euclidean distance is used in our current model to compute the dissimilarity between vectors.

The framework allows different ways to encode the frequencies of occurrence: simple *flat counts of occurrences*, the *TF-IDF weighting scheme*, and the *word category histograms* (*WCH*). One of these representations, i.e. the TF-IDF (“term (or word) frequency times inverse document frequency”) weighing scheme is commonplace for text document classification settings, which is also the case for our model.

The third method from the enumeration above encodes the vector representation as a *word category histogram* (*WCH*). In order to compute such histograms, first a SOM map [8] is trained having the global vocabulary of words as input data space to arrive at a *reduced set of semantic categories of words*. Words with similar meaning are clustered together by the unsupervised SOM neural network. In this SOM training, the words are represented as vectors of frequencies of their occurrence in the different concept documents. Equally like the document vectors, the word vectors are collected from the same word/document occurrence matrix, but after transposing this matrix.

For a given document, by summing together the frequencies in document of all the words that belong to one and the same category, and merely keeping these summed frequencies of the words per categories as histogram vector features, we arrive at a reduced dimensionality for the document vector representation. In other words, the WCH histogram is obtained by counting the occurrences for words of distinct categories instead of counting the occurrences of distinct words. The categories of words are in fact semantic categories of words, as induced by training the SOM map on the entire global vocabulary of words used in the concept documents. By reducing the dimensionality of the feature vectors, the word category histograms also *reduce the data sparseness* of the vector representations in our setting.

The vector representations are sparse, since the concept documents are small, and consequently each document contains only a few of the global vocabulary of words. For the rest of the words, the document vector has value zero as the feature corresponding to a word that is absent from the document. Besides using the WCH histograms, two other ways of reducing the number of zeros in our vector representations are the *centroid vector* and the *category vector* [10, 2]. In our current experiments, they are only used for reducing the data sparseness of the ontology 2 concept documents, in order to improve the semantic discrimination power of taxonomy 2, which plays the role of a decision tree. We have used the idea of centroid in the following way: the average vector of the vector representations of all the concepts in the sub-tree rooted by the given concept, including the root itself. Likewise, the category vector of a sub-tree is the sum of all the concept vectors of the sub-tree, normalized to unit length (unit norm).

V. Experimental Results

We have evaluated our ontology matching approach with the Ontology Alignment Evaluation Initiative[1] (OAEI) benchmark data set for the bibliography domain [5, 3]. The set consists of a suite of ontologies obtained by altering an initial ontology. The alterations have been made along with different aspects, such as the class and property names, removing the properties or the instances, distorting the specialization hierarchy (i.e. the taxonomy, the subclass relations) etc. The result is a set consisting of more than 50 pairs of ontologies to be matched, where ontology 1 is one of the altered ontologies and ontology 2 is the initial ontology.

A. Experimental Setup

The initial ontology from the OAEI benchmark suite of ontologies consists of 33 concepts. 24 of them are established into the *Reference* taxonomy, having the concept *Reference* in the root. The remaining concepts are called "special classes" by the authors of the benchmark suite. They are arranged in very small taxonomies or alone, playing rather the role of range concepts for the relations which start (as domain) from the 24 concepts in the *Reference* taxonomy. The concept *Reference* is the most generic concept in the bibliography topic, having the meaning of "any bibliographic reference". Figure 3 illustrates this *Reference* taxonomy, which always plays the role of "ontology 2" (initial ontology) in the experiments described below. In all of our experiments we classify the document-style "ontology 1" concepts against the *Reference* taxonomy, where ontology 1 is one of the altered ontologies.

For the experimental evaluation, we assess the mapping from ontology 1 concepts to ontology 2 concepts. We evaluate the correctness of the pairs (*o*1*_concept*, *o*2*_concept*) by comparison to a *reference alignment*. As being a benchmark suite for evaluating the ontology matching algorithms, the OAEI bibliography dataset comes equipped with a reference alignment for each of the pairs of ontologies to be matched. A reference alignment is a list of concept pairs where the correctness of the pairs has been found consensually by people.

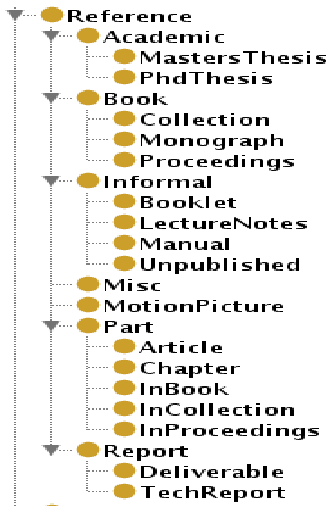

Figure 3. The *Reference* taxonomy - "ontology *o*2".

B. Evaluation Measures

The most important measure proposed in the literature for evaluating the ontology matching systems is the pair consisting of *precision* and *recall*. Their computation is based on counting the correct and the incorrect pairs of concepts from the matched ontologies. This is an "all-or-nothing" measure, and the alignments found by an ontology matching system can nevertheless contain *near misses*, i.e. pairs of concepts (*o*1*_concept*, *o*2*_concept*) that are semantically close to a correct pair. This observation led to the *relaxed precision* and *recall*, which are weighted by a measure of *overlap proximity* [5]. One of the overlap proximities proposed is the *symmetric* one [5], which weights the precision and recall by the similarity between the concept from the alignment found by the system *S* and the concept from the reference alignment *R*. The similarity between two concepts is understood here as a taxonomic similarity. It is inverse proportional with the taxonomic distance between the concepts in terms of the number of taxonomy edges which separate the two concepts in the ontology the concepts belong to. This ontology can be either of the two ontologies

1 http://oaei.ontologymatching.org/

*o*1 and *o*2 being matched, so, symmetrically, both elements *o*1*_concept* and *o*2*_concept* of the pairs are compared in the system alignment *S* versus the reference alignment *R*.

We propose here an evaluation measure inspired from the area of semantic classification (like ontology enrichment and named entity classification): the *learning accuracy* [6, 2, 10, 1, 13]. By choosing this measure, we consider correct semantic classifications with different levels of detail. For instance, the named entity *Tom* can be mapped to the concept *cat*, *felin*e, *carnivore*, *mammal*, or *animal* with different levels of detail, as a consequence of different hypernym-hyponym taxonomic distances between the concept chosen by the system and the correct one. Consequently, being a "near miss" measure (as opposed to "all-or-nothing"), and also by weighting the semantic closeness by the same measure of taxonomic distance, the learning accuracy is in agreement with the relaxed precision and recall proposed in the ontology matching literature. The only difference is that, somehow non-symmetrically, we measure the taxonomic similarity between the system alignment *S* and the reference alignment *R* only in what concerns *o*2*_concept*, so only for the second component of the pairs in the two compared alignments *S* and *R*. This is because our ontology matching system classifies semantically any given (fixed) *o*1*_concept* against the ontology *o*2.

For a given concept *o*1*_concept*, let *s* be the concept (from ontology *o*2) assigned by the system, and *r* be the correct concept according to the reference alignment. Then the learning accuracy is the average over all the classified concepts *o*1*_concept* (from ontology *o*1) of the function *LA*(*s*, *r*), where the function *LA* is defined as in [6]:

$$LA(s, r) = \frac{\delta(top, a) + 1}{\delta(top, a) + \delta(a, s) + \delta(a, r) + 1} \quad (1)$$

Where *top* is the root of the taxonomy, and *a* is the least common subsumer of the concepts *s* and *r* (i.e. the most specific common hypernym of *s* and *r*). $\delta(x, y)$ is the taxonomic distance between concepts *x* and *y*. The learning accuracy has a real value between 0 and 1, also interpreted as a percentage between 0 and 100%.

In the experimental results reported in this paper, the *symmetric learning accuracy* actually corresponds to the definition in formula (1), and the *learning accuracy* is a historically initial version of the learning accuracy measure introduced by [7] and also defined in [6]:

$$LA'(s, r) = \frac{\delta(top, a) + 1}{\delta(top, r) + 1} \quad \text{if } s \text{ is ancestor of } r \text{ (then also } a = s)$$

$$LA'(s, r) = \frac{\delta(top, a) + 1}{\delta(top, a) + 2 * \delta(a, s) + 1} \quad \text{otherwise} \quad (2)$$

We evaluated our experiments in terms of both variants of the *learning accuracy* and also a third variant of it, which is called *edge measure*. The *edge measure* counts literally the taxonomic distance between the system predicted concept *s* (i.e. according to the system alignment) and the one from the reference alignment *r*.

C. Evaluation Results

In the experiments reported in tables I to IV, we use the following notations. *Flat* means document vectors represented as flat counts of occurrences. When *TFIDF* occurs together with *WCH*, then the TF-IDF weighting scheme is first applied to the flat count document vectors. Then the result TF-IDF vectors are converted into WCH histograms, thus reducing the concept vector dimensionality. *WHCM* means "big histograms", i.e. the vector dimensionality is about 100 for the different experimental runs, whereas *WCHm* means "small histograms", and the vector dimensions are only about 35. *Centr* and *categ* are the means to compute the vector of a taxonomic node of ontology *o*2, as a centroid or as a category of all the nodes in the subtree of the concept node.

Table I illustrates the results of matching the initial ontology (playing the role of ontology 2) with an ontology (acting as ontology 1) altered by substituting the concept names with synonyms, e.g. *Unpublished* became *Manuscript*, and *LectureNotes* became *CourseMaterial*. This pair of ontologies is the test nr. 205 in the OAEI benchmark suite and the test pair is equipped (built-in in the dataset) with the following reference alignment:

$$R_{205} = \{ (Manuscript, Unpublished), (CourseMaterial, LectureNotes), \ldots \} \quad (3)$$

In the altered ontology of the test pair in Table II all the properties and relations have been removed from the initial ontology. In table III ontology 1 from the test pair has been altered by completely translating the initial ontology (except the comments) in French. *Unpublished* becomes *NonPublié* and *LectureNotes* becomes *Polycopié*. Table IV shows a test pair in which the altered ontology has an enriched taxonomic structure as compared to the initial ontology. The altered *Reference* taxonomy *o*1 has 45 concepts compared to 24 of the initial *Reference* taxonomy *o*2 (see Figure 4 versus Figure 3).

Across the three tests, there is a natural tendency that using the TF-IDF weighting measure improves the results. TF-IDF is a semantic-oriented scheme, which gives a higher weight to the words appearing in fewer documents. This leads to an increase in the semantic discrimination power between documents (actually between concepts in our setting, since the concepts are represented by documents). At the same time, there is also a slight improvement when representing ontology 2 concepts as category, as compared to representing them as centroid.

TABLE I. ALIGNMENT 205 – SYNONYMS

Experiment	Symmetric Learning Accuracy	Learning Accuracy	Edge Measure
flat, centr	0.329861	0.368055	2.458333
flat, WCHM, centr	0.315972	0.343055	2.333333
flat, WCHm, centr	0.430555	0.423611	1.583333
TFIDF, centr	0.427083	0.430555	1.625
TFIDF, WCHM, centr	0.40972	0.40278	1.66667
TFIDF, WCHm, centr	0.45833	0.46667	1.75
flat, categ	0.33681	0.35139	2.70833
flat, WCHM, categ	0.37847	0.39583	2.16667
flat, WCHm, categ	0.38125	0.39861	2.16667
TFIDF, categ	**0.5**	**0.5**	**1.375**
TFIDF, WCHM, categ	0.43403	0.4375	1.70833
TFIDF, WCHm, categ	0.48958	0.49444	1.54167

TABLE II. ALIGNMENT 228 – NO PROPERTY

Experiment	Symmetric Learning Accuracy	Learning Accuracy	Edge Measure
flat, centr	0.30903	0.35556	2.58333
flat, WCHM, centr	0.30903	0.35556	2.58333
flat, WCHm, centr	0.37847	0.38889	1.91667
TFIDF, centr	0.5625	0.5625	1.20833
TFIDF, WCHM, centr	0.40972	0.40972	1.625
TFIDF, WCHm, centr	0.38542	0.40972	2
flat, categ	0.39792	0.39444	2.54167
flat, WCHM, categ	0.39583	0.43056	2.25
flat, WCHm, categ	0.39722	0.4125	2.29167
TFIDF, categ	**0.76389**	**0.72917**	**0.79167**
TFIDF, WCHM, categ	0.59792	0.59444	1.33333
TFIDF, WCHm, categ	0.38681	0.40417	2.08333

The very best results attained (76.4%) are in the case when the properties are suppressed, whereas the second best accuracy corresponds to the expanded taxonomy (63.2%). This is not a surprise, since in both cases the altered ontology keeps the concept names the same as in the initial ontology. Finally, the worst accuracy (a maximum of 47.2%) is obtained when the alteration means a translation to French. In this case the semantic classifier, which is the heart of our ontology matcher, had to face the cross language problem.

TABLE III. ALIGNMENT 206 – FOREIGN NAMES

Experiment	Symmetric Learning Accuracy	Learning Accuracy	Edge Measure
flat, centr	0.35764	0.35556	1.91667
flat, WCHM, centr	0.35417	0.36806	2.08333
flat, WCHm, centr	0.375	0.36806	1.75
TFIDF, centr	0.38889	0.38194	1.66667
TFIDF, WCHM, centr	0.39931	0.39028	1.70833
TFIDF, WCHm, centr	0.4375	0.4375	1.54167
flat, categ	0.32431	0.33472	2.20833
flat, WCHM, categ	0.38542	0.41667	2.08333
flat, WCHm, categ	0.38889	0.40278	1.95833
TFIDF, categ	0.43056	0.43056	1.58333
TFIDF, WCHM, categ	0.42014	0.41806	1.66667
TFIDF, WCHm, categ	**0.46528**	**0.47222**	**1.54167**

TABLE IV. ALIGNMENT 238 – EXPANDED HIERARCHY

Experiment	Symmetric Learning Accuracy	Learning Accuracy	Edge Measure
flat, centr	0.37153	0.40278	2.33333
flat, WCHM, centr	0.43403	0.44444	1.875
flat, WCHm, centr	0.41667	0.43056	1.95833
TFIDF, centr	0.58333	0.56944	1.16667
TFIDF, WCHM, centr	0.51042	0.50694	1.41667
TFIDF, WCHm, centr	0.48056	0.48056	1.66667
flat, categ	0.38889	0.40556	2.45833
flat, WCHM, categ	0.46806	0.46944	2.04167
flat, WCHm, categ	0.39583	0.43056	2.25
TFIDF, categ	0.57639	0.5625	1.20833
TFIDF, WCHM, categ	**0.63194**	**0.62639**	**1.16667**
TFIDF, WCHm, categ	0.47292	0.49861	2.04167

VI. CONCLUSIONS AND FURTHER WORK

We have presented an unsupervised top-down neural network based model for class based ontology matching. The matching is cast as a document classification problem, where the concepts of the two ontologies being matched are considered as text documents. The matcher provides good alignment results, which means a reduced effort required from a user assistant to fix the alignments found by the system.

As future work, we will use other data sets from the Ontology Alignment Evaluation Initiative, such as real life expressive ontologies in the anatomy domain, or large thesauri in the agriculture domain. For the sake of symmetry

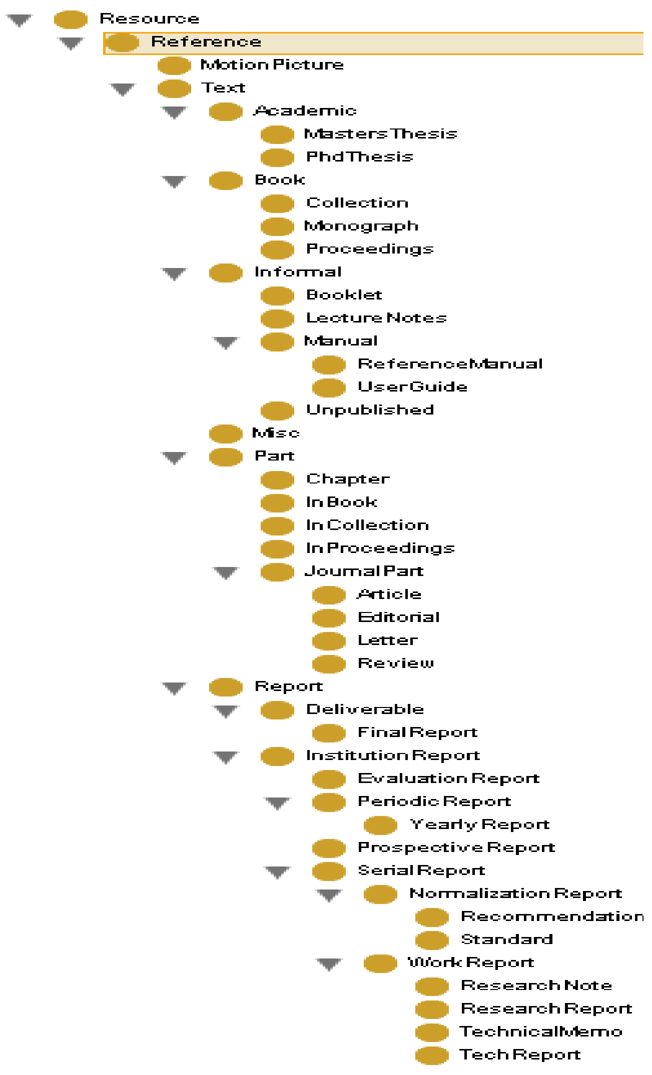

Figure 4. The enriched *Reference* taxonomy from OAEI benchmark test pair nr. 238.

and a perfect compliance with the relaxed precision and recall used in the literature [5], we will interchange the role of "ontology 1" and "ontology 2" as defined in our setting. We will compute the learning accuracy of classifying ontology 1 concepts against the taxonomy of ontology 2, then interchange the two ontologies and compute again the learning accuracy. The final measure for the quality of matching will be the average of the learning accuracy of the two symmetric alignments found by the matcher.

ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers for the very useful comments. Part of this work was supported by the grant ID_170/672 from the National Research Council of the Romanian Ministry for Education and Research.

REFERENCES

[1] E. Alfonseca and S. Manandhar, "Extending a lexical ontology by a combination of distributional semantics signatures", in A. Gómez-Pérez, V.R. Benjamins, eds., *13th EKAW Conference, LNAI*, Springer, 2002, pp. 1-7.

[2] P. Cimiano and J. Völker, 2005, "Towards large-scale, open-domain and ontology-based named entity classification", *RANLP'05 Conference*, 2005, pp. 166-172.

[3] J. David and J. Euzenat, "Comparison between ontology distances (preliminary results)", in A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, K. Thirunarayan, eds., *The semantic web, Lecture notes in computer science 5318*, 2008, pp. 245-260.

[4] M. Dittenbach, D. Merkl, and A. Rauber, "Organizing and exploring high-dimensional data with the Growing Hierarchical Self-Organizing Map", in L. Wang, et al., eds., *1st International Conference on Fuzzy Systems and Knowledge Discovery*, vol. 2, 2002, pp. 626-630.

[5] J. Euzenat and P. Shvaiko, *Ontology Matching*, Springer-Verlag, Heidelberg, 2007.

[6] M. Grobelnik, P. Cimiano, E. Gaussier, P. Buitelaar, B. Novak, J. Brank, and M. Sintek, "Task description for PASCAL challenge. Evaluating ontology learning and population from text", 2006.

[7] U. Hahn and K. Schnattinger, "Towards text knowledge engineering", *15th AAAI Conference and 10th IAAI Conference*, 1998, pp. 524-531.

[8] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela, "Self-organization of a massive document collection", *IEEE Transactions on Neural Networks*, **11**, 3, 2000, pp. 574-585.

[9] R. Levy, J. Henriksson, M. Lyell, X. Liu, and M.J. Mayhew, "Ontology matching across domains", *Workshop on Agent-based Technologies and applications for enterprise interOPerability, AAMAS Conference*, 2010.

[10] V. Pekar and S. Staab, "Taxonomy learning – factoring the structure of a taxonomy into a semantic classification decision", *COLING'02 Conference,* 2002, pp.786-792.

[11] M. Sabou, J. Gracia, S. Angeletou, M. d'Aquin, and E. Motta, "Evaluating the Semantic Web: a task-based approach", The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, Busan, Korea, 2007, pp. 423-437.

[12] K. Sivashanmugam, K. Verma, A. Sheth, J. Miller, "Adding semantics to Web services standards", *ICWS03 Conference*, 2003.

[13] H. F. Witschel, "Using decision trees and text mining techniques for extending taxonomies", *Learning and Extending Lexical Ontologies by using Machine Learning Methods, Workshop at ICML-05*, 2005, pp. 61-68.