HAL open science

Mining Concept Similarities for Heterogeneous Ontologies

Konstantin Todorov, Peter Geibel, Kai-Uwe Kühnberger

To cite this version:

Konstantin Todorov, Peter Geibel, Kai-Uwe Kühnberger. Mining Concept Similarities for Heterogeneous Ontologies. ICDM 2010 - 10th Industrial Conference on Data Mining, Jul 2010, Berlin, Germany. pp.86-100, 10.1007/978-3-642-14400-4_7 . hal-01987787

HAL Id: hal-01987787

https://hal.science/hal-01987787v1

Submitted on 23 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mining Concept Similarities for Heterogeneous Ontologies

Konstantin Todorov[1], Peter Geibel[2], and Kai-Uwe Kühnberger[3]

[1] Laboratory MAS, École Centrale Paris, F-92 295 Châtenay-Malabry, France
[2] TU Berlin, Fakultät IV, Franklinstr. 28/29, 10587 Berlin, Germany
[3]Institute of Cognitive Science, Universtity of Osnabrück, Germany

Abstract. We consider the problem of discovering pairs of similar concepts, which are part of two given source ontologies, in which each concept node is mapped to a set of instances. The similarity measures we propose are based on learning a classifier for each concept that allows to discriminate the respective concept from the remaining concepts in the same ontology. We present two new measures that are compared experimentally: (1) one based on comparing the sets of support vectors from the learned SVMs and (2) one which considers the list of discriminating variables for each concept. These lists are determined using a novel variable selection approach for the SVM. We compare the performance of the two suggested techniques with two standard approaches (Jaccard similarity and class-means distance). We also present a novel recursive matching algorithm based on concept similarities.

1 Introduction

In A[rtificial] I[ntelligence], an ontology, in the broadest sense, is understood as a collection of *concepts* and *relations* defined on these concepts, which altogether describe and structure the knowledge in a certain domain of interest. The *O[ntology] M[atching]* problem stems from the fact that different communities, independently from one another, are likely to adopt different ontologies, given a certain domain of interest. In consequence, multiple *heterogeneous* ontologies, describing similar or overlapping fractions of the world are created. An ontology matching procedure aims at reducing this heterogeneity by yielding assertions on the relatedness of cross-ontology concepts, in an automatic or semi-automatic manner. To these ends, according to [5], a matching procedure commonly relies on extensional (related to the concepts instances), structural (related to the inter-ontology concepts relations), terminological (language-related) or semantic (related to logical interpretation) information, separately or in combination.

In this paper, we will expand on one of these general types of ontology matching, known as *instance-based* matching. This comprises a set of approaches for measuring the similarity of concepts from two source ontologies based on their extensions – the instances that populate the respective concepts [9]. We consider two training sets with classified examples, one for each of the two source ontologies. We assume that the examples in each training set are described using the

same set of variables or attributes. The classes of the examples, however, belong to two separate conceptual systems that serve as taxonomies and possess a hierarchical, tree-like structure. The ontology nodes are thus implicitly assigned the relevant examples from the training set, taking into account that the hierarchical structure represents an *is_a*-relationship between concept nodes.

Being given these two training sets together with the source ontologies, we now consider the *data mining task* of discovering similarities between concepts of the source ontologies based on the instances and the structures of the ontologies. In the case that the similarities imply a suitable one-to-one mapping between concept nodes of the two source ontologies, they can be used for computing the "intersection" of the two source ontologies which forms the result of the matching process.

In contrast to other learning-based matching approaches, which aim at estimating joint probabilities for concept pairs [4], we compute the similarity of two concepts based on the similarity of their intra-ontology classifiers. In the case of the S[upport] V[ector] M[achines] [3], this can be achieved by comparing the support vectors characterizing the respective concepts. The support vectors are examples for the respective concept that can be considered important for discriminating it from other classes in the same ontology, and are thus relevant for characterizing it with respect to the whole ontology.

We present a second generic approach, which bases concept similarity on variable selection techniques that capture characteristics of the data in terms of the relevance of variables for classifier learning. In the case of text documents, these approaches allow to determine those words or terms that discriminate the respective concept from other concepts in the same ontology. We introduce a new technique for selecting variables for SVMs and use it for determining the similarity of two concepts by comparing their lists of discriminative attributes. The two novel similarity measures are tested against two standard techniques used in state-of-the-art approaches: the Jaccard similarity and the class-means distance.

The remainder of the paper has the following structure. Section 2 presents relevant related approaches to ontology matching. Section 3 sets the ontology matching framework in terms of definitions and assumptions. The two new approaches to measure concept similarity are suggested further in Section 4 (comparison of support vectors) and Section 5 (variable selection). A variable selection criterion for SVM, together with a short introduction to the classifiers is presented in Section 6. Section 7 descriebs a recursive matching procedure based on the proposed similarity measures. Finally, we present our experimental results in Section 8 before we conclude with Section 9.

2 Related Work

As mentioned in the introduction, an important part of the existing OM-approaches, including ours, are characterized as *extensional*, i.e. grounded in the external world, relying on instances in order to judge intensional similarity.

Among the basic assumptions of such approaches is that two ontologies use the same instances to populate different conceptual structures and when this is not so, mechanisms for extracting instances (from text corpora or other external sources) should be made available (FCA-MERGE [15]). Other techniques rely on estimating the concepts similarity by measuring class-means distances (CAIMAN [11]) or estimating joint probabilities by the help of machine learning techniques (GLUE [4]). Most of these standard approaches are based on rather restrictive assumptions, tend to be costly on a large scale or perform well for leaf-nodes but fail to capture similarities on higher levels.

It is a relatively old idea that the gap between two conceptual systems can be bridged by using the relations of the concepts within each of the systems. The use of structure for judging concept similarities has found response in the OM community, some examples of such algorithms being ANCHOR-PROMPT [13], ABSURDIST [6] and ONION [12].

Our work is much in line with the tradition of extensional concept representation and similarity measurement. The relations between concepts are used in order to improve and optimize the extensional similarity judgments. They are taken into account by the recursive matching algorithm that is based on pairwise concept similarities. In technical terms, an advantage of our method is that most of it is accomplished with the training phase of the classification task. In contrast to most instance-based techniques, our matching approach does not rely on intersections of instance sets, nor on the estimation of joint probabilities. It works with instance sets that might be different for both ontologies, which avoids taking the costly step of extracting instances from external sources. Finally, in case of textual instances, the method makes available the list of the most important words that characterize a similar pair of concepts - information not readily available in the approaches cited above.

3 Populated Ontologies: Definition and Assumptions

Throughout this paper, an ontology, whose concepts are labels of real-world instances of some kind, will be defined in the following manner (modifying a definition found in [15]).

Definition 1. *A* ***populated ontology*** *is a tuple* $O = \{C, \texttt{is_a}, R, I, g\}$, *where* C *is a set whose elements are called concepts,* `is_a` *is a partial order on* C, R *is a set of other (binary) relations holding between the concepts from the set* C, I *is a set whose elements are called instances and* $g : C \to 2^I$ *is an injection from the set of concepts to the set of subsets of* I.

In the formulation above, a concept is *intensionally* defined by its relations to other concepts via the partial order and the set R, and *extensionally* by a set of instances via the mapping g. We note that the sets C and I, are compulsorily non-empty, whereas R can be the empty set. In view of this remark, the definition above describes a *hierarchical ontology*: an ontology which, although not limited to subsumptional relations, necessarily contains a hierarchical backbone, defined

by the partial order on the set of concepts. If non-empty, R contains relations, defined by the ontology engineer (for instance, `graduated_at`, `employed_by`, etc.). The set I is a set of concept instances – text documents, images or other (real world data) entities, representable in the form of real-valued vectors. The injection g associates a set of instances to every concept. By definition, the empty set can be associated to a concept as well, hence not every concept is expected or required to have instances. Whether g takes inheritance via subsumption into account in defining a concept's instance-set (hierarchical concept instantiation) or not (non-hierarchical instantiation) is a semantics and design-related issue [9].

Let us consider two ontologies O_1 and O_2 and their corresponding instance-sets $I_1 = \{\mathbf{i}_1^1, ..., \mathbf{i}_{m_1}^1\}$ and $I_2 = \{\mathbf{i}_1^2, ..., \mathbf{i}_{m_2}^2\}$, where each instance is represented as an n-dimensional vector and m_1 and m_2 are integers. For a concept $A \in C_1$ from ontology O_1, we define a labeling $S^A = \{(\mathbf{i}_j^1, y_j^A)\}$, where y_j^A takes a value $+1$ when the corresponding instance $\mathbf{i}_j^1$ is assigned to A, and -1 otherwise, for $j = 1, ..., m_1$. The labels split the instances of O_1 into those that belong to the concept A (positive instances), and those that do not (negative ones) defining a binary classification training set. The same representation can be acquired analogously for any concept in both ontologies O_1 and O_2.

4 Concept Similarity via Comparison of Intra-Ontlogy Classifiers

A straightforward idea for determining the similarity $sim(A, B)$ of two concepts A and B consists in comparing their instance sets $g(A)$ and $g(B)$. For doing so, we thus need a similarity measure for instances $\mathbf{i}^A$ and $\mathbf{i}^B$. We can use, for instance, the scalar product and the cosine $s(\mathbf{i}^A, \mathbf{i}^B) = \frac{\langle \mathbf{i}^A, \mathbf{i}^B \rangle}{\|\mathbf{i}^A\| \|\mathbf{i}^B\|}$ (i.e. the normalized scalar product). Based on this similarity measure for elements, the similarity measure for the sets can be defined by computing the similarity of the mean vectors corresponding to class prototypes, i.e.

$$sim_{proto}(A, B) = s\Big(\frac{1}{|g(A)|} \sum_{j=1}^{|g(A)|} \mathbf{i}_j^A, \frac{1}{|g(B)|} \sum_{k=1}^{|g(B)|} \mathbf{i}_k^B\Big) . \tag{1}$$

This method underlies the Caiman approach [11] in which concepts are assumed to be represented by their mean vector. While this approach might be suitable for leaf nodes, in which the data might be characterized by a unimodal distribution, it is generally bound to fail for nodes higher up in the tree, whose instance set might be composed of several subsets resulting in a multi-modal distribution.

The theory of hierarchical clustering (e.g., [1]) provides alternative methods for defining similarities for pairs of sets. Examples are the similarity measures

$$sim_{min}(A, B) = \min_{j,k} s(\mathbf{i}_j^A, \mathbf{i}_k^B), \quad sim_{max}(A, B) = \max_{j,k} s(\mathbf{i}_j^A, \mathbf{i}_k^B) \tag{2}$$

and

$$sim_{avg}(A, B) = \frac{1}{|g(A)||g(B)|} \sum_{j,k} s(\mathbf{i}_j^A, \mathbf{i}_k^B) . \tag{3}$$

The three measures correspond to different types of clustering methods: complete link, single link, and average link clustering.

It is well-known that computing the similarity based on these measure can be quite complex if the training sets are large. Moreover, the min- and max-based measures can get easily spoiled by outliers in the training sets, whereas the average link approach is also prone to the problem of multi-modal distributions.

Ontologies are based on the idea of discriminating between different concepts or classes in order to conceptualize a domain. This idea also forms the basis of the SVM described in more detail in Section 6. After successfully learning a classifier, the support vectors of class A correspond to those examples in $g(A)$ that have turned out to be most important for discriminating it from the other class containing the instances for the concepts in $C_1 \setminus \{A\}$. This means in particular that the support vectors for the A-classifier are elements of $g(A)$, whereas those for B can be found in $g(B)$. The idea which we propose is to base the measures in (2) and (3) only on the support vectors. If we can train the classifiers successfully (i.e., with a low error), this will reduce complexity and can help solve the problem of outliers and irrelevant examples.

5 Concept Similarity via Variable Selection

V[ariable] S[election] techniques (reviewed in [7]) serve to rank the input variables of a given problem (e.g. classification) by their importance for the output (the class affiliation of an instance), according to certain evaluation criteria. Technically speaking, a VS procedure assigns to each variable a real value – *a score* – which indicates the variable's pertinence. This can be of help for dimensionality reduction and for extracting important input-output dependencies. Assuming that instances are represented as real-valued vectors, a VS procedure in our study indicates which of the vector dimensions are most important for the separation of the instances (within a single ontology) into those that belong to a given concept and those that do not. In the case of documents, these might be words or tokens that distinguish the respective concept from others in the same ontology.

By the help of variable selection procedures carried out independently for two concepts $A \in C_1$ and $B \in C_2$, on their corresponding sets $S^A = \{(\mathbf{i}_j^1, y_j^A)\}$, $j = 1, ..., m_1$ and $S^B = \{(\mathbf{i}_k^2, y_k^B)\}$, $k = 1, ..., m_2$, one scores the input variables by their importance for the respective class separations. In that, the concepts A and B can be represented by the lists of their corresponding variables scores in the following manner:

$$Scores(A) = (s_1^A, s_2^A, ..., s_n^A),\ Scores(B) = (s_1^B, s_2^B, ..., s_n^B), \tag{4}$$

Note that to score the input variables, one could rely on various selection techniques. In previous studies, we have tested structural dimension reducing methods (discriminant analysis), standard feature selection techniques for text categorization (point-wise mutual information, chi-square statistics and document frequency thresholding), as well as an SVM-based method [16]. The latter

falls in the focus of the current paper and will be, therefore, introduced in more detail in the following section.

On the basis of the concept representations in (4), different measures of concept similarity can be computed [16]. The k-TF measure looks for re-occurring elements in two lists of top $k-$scored variables. Alternatively, parameter-free measures of statistical correlation, which act as measures of similarity, can be computed over the ranks or directly over the scores associated to the variables. Pearson's coefficient, which has been used in the experimental part of this paper, is given by

$$r = \frac{\sum_{i=1}^{n}(s_i^A - s_{mean}^A)(s_i^B - s_{mean}^B)}{\sqrt{\sum_{i=1}^{n}(s_i^A - s_{mean}^A)^2}\sqrt{\sum_{i=1}^{n}(s_i^B - s_{mean}^B)^2}}, \tag{5}$$

where s_{mean}^A and s_{mean}^B are the means of the two respective score lists over all n variables.

6 A Variable Selection Method for the SVM

In the following, Section 6.1 aims at familiarizing the reader with several concepts from the SVM theory that are relevant for the introduction of our SVM-based variable selection criterion described, in turn, in Section 6.2.

6.1 Support Vector Machines

The SVMs are inductive machine learners, initially designed to solve binary classification tasks [3]. For reasons of space, we will provide knowledge about the method limited to what is sufficient to understand the ideas behind SVM-based variable selection (comprising existing methods and our approach).

Let us consider the following binary classification layout. Assume we have l observations $\mathbf{x}_i \in \mathbb{R}^n$ and their associated "truth" $y_i \in \{-1, 1\}$. Data are assumed to be i.i.d. (independent and identically distributed), drawn from an unknown probability distribution $P(\mathbf{x}, y)$. The goal of binary classification is to "learn" the mapping $\mathbf{x}_i \mapsto y_i$ which is consistent with the given examples. Let $\{f(\mathbf{x}, \sigma)\}$ be a set of such possible mappings, where σ denotes a set of parameters. Such a mapping is called a classifier and it is deterministic - for a certain choice of $\mathbf{x}$ and σ it will always give the same output f.

The **actual risk**, or the expectation of the test error for such a learning machine is

$$R(\sigma) = \int \frac{1}{2}|y - f(\mathbf{x}, \sigma)| dP(\mathbf{x}, y).$$

The quantity $1/2|y - f(\mathbf{x}, \sigma)|$ is called *the loss*. Based on a finite number, l, of observations, we calculate the **empirical risk**

$$R_{emp}(\sigma) = \frac{1}{2l}\sum_{i=1}^{l}|y_i - f(\mathbf{x}_i, \sigma)|,$$

which is a fixed number for a given training set $\{(\mathbf{x}_i, y_i)\}$ and a certain choice of parameters σ.

For losses taking values 0 or 1, with probability $1-\eta$, $0 \leq \eta \leq 1$, the two risks are related in the following manner:

$$R(\sigma) \leq R_{emp}(\sigma) + \sqrt{\frac{h\log(\frac{2l}{h}) + 1 - \log(\frac{\eta}{4})}{l}}, \tag{6}$$

where h is a nonnegative integer which will play a core role in our variable selection procedure, called the *VC dimension*. The bound (6) gives an insight in one very important aspect of generalization theory of statistical learning. The term $\sqrt{\frac{hlog(\frac{2l}{h})+1-log(\frac{\eta}{4})}{l}}$, called *VC confidence* is "responsible" for the *capacity* of the learner, i.e. its ability to classify unseen data without error. The other right-hand quantity in (6) - the empirical risk, measures the *accuracy* attained on the particular training set $\{(\mathbf{x}_i, y_i)\}$. What is sought for is a function which minimizes the bound on the actual risk and thus provides a good balance between capacity and accuracy - a problem known in the literature as *capacity control*.

The presented risk bound does not depend on $P(\mathbf{x}, y)$ and it can be easily computed provided the knowledge of h. We introduce what does this parameter stand for. Let us consider the set of functions $\{f(\mathbf{x}, \sigma)\}$ with $f(\mathbf{x}, \sigma) \in \{-1, 1\}, \forall \mathbf{x}, \sigma$. In a binary classification task there are 2^l possible ways of labeling a set of l points. If for each labeling there can be found a member of $\{f(\sigma)\}$ which correctly assigns these labels, we say that the given set of points is *shattered* by the given set of functions. The VC dimension is a property of such a family of functions, which is defined as the maximum number of training points that can be shattered by that family. Although in general difficult to compute directly, an upper bound for the VC-dimension can be computed depending on the weight vector $\mathbf{w}$ and on properties of the data. In the SVM*light* implementation, which we have used for our experiments in Section 8.4, the VC dimension is estimated based on the radius of the support vectors [10].

Now, let us return to binary classification. Consider the input space $X \subseteq \mathbb{R}^n$ and the output domain $Y = \{-1, 1\}$ with a training set $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l)\} \in (X, Y)^l$. SVM is a linear real function $f : X \to \mathbb{R}$ with

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b,$$

where $\sigma = (\mathbf{w}, b) \in \mathbb{R}^n \times \mathbb{R}$. The separating hyperplane in the input space X is defined by the set $\{\mathbf{x} | f(\mathbf{x}) = 0\}$. The decision rule assigns an input vector $\mathbf{x}$ positive if and only if $f(\mathbf{x}) \geq 0$ and negative - otherwise. (The inclusion of 0 in the first case and not in the second is conventional.)

We are looking for the best decision function $f(\mathbf{x})$ which separates the input space and maximizes the distance between the positive and negative examples closest to the hyperplane. The parameters of the desired function are found by solving the following quadratic optimization problem:

$$\min_{\mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}} \frac{1}{2} \|\mathbf{w}\|^2$$

under the linear constraints

$$\forall i = 1, ..., l,\ y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle) + b) \geq 1.$$

When data are not linearly separable in the input space, they are mapped into a (possibly higher dimensional) space, called *feature space* where a linear boundary between both classes can be found. The mapping is done implicitly by the help of a kernel function which acts as a dot product in the feature space.

When solving the above optimization problem, the weight vector $\mathbf{w}$ can be expressed as a linear combination of the input vectors. It turns out that only certain examples have a weight different from 0. These vectors are called *support vectors* for they define the separating hyperplane and can be considered the examples closest to it.

6.2 VC-dimension-Based Variable Selection

SVM-based variable selection has already been studied in the past couple of years. Guyon *et al.* proposed the SVM-RFE algorithm [8] for selecting genes which are relevant for cancer classification. The removal criterion for a given variable is minimizing the variation of the weight vector $\| \mathbf{w} \|^2$, i.e. its sensitivity with respect to a variable. Rakotomamonjy *et al.* carried out experiments for pedestrian recognition by the help of a variable selection procedure for SVMs based on the sensitivity of the margin according to a variable [14]. A method based on finding the variables which minimize bounds on the *leave-one-out* error for classification was introduced by Weston *et al.* [17]. Bi *et al.* developed the VS-SSVM variable selection method for regression tasks applied to molecules bio-activity prediction problems [2].

The variable selection criterion that we propose is based on the sensitivity of the VC dimension of the SVM classifiers with respect to a single variable or a block of variables. As we have seen in the previous subsection, for different values of the VC dimension h, different values of the VC confidence (describing the capacity of the classifier) will be computed and thus different bounds on the actual risk (6), where from the generalization power of the classifier will change. Our main heuristics can be formulated as *"a less informative variable is one, which the VC confidence of the classifier is less sensitive to"*.

For computational reasons the evaluation function of our variable selection procedure will be formulated in terms of VC dimension directly, instead of in terms of the VC confidence. This is plausible since the VC confidence is monotonous in h. Thus, the i-th variable is evaluated by

$$eval_i = h(H) - h(H^{(i)}),\ i = 1, ..., n, \tag{7}$$

where $h(H)$ is the VC dimension of a set of SVM hypotheses H constructed over the entire data set and $h(H^{(i)})$ is the same quantity computed after the removal of the i-th variable (whose pertinence is to be evaluated) from the data set.

Tests of the performance of suggested variable evaluation criterion are presented in [16].

7 A Similarity-Based Matching Procedure

Fig. 1 shows two ontologies that intend to organize news articles in different structurally related topics. This example will help us introduce (in the current section) and evaluate (in the following section) a recursive ontology matching procedure. We abstract from the concept names being similar, for they are not taken into consideration in the concept similarity measurement. We have populated the concepts of the ontologies with documents taken from the 20 Newsgroups[1] dataset in a way that any two document sets across both ontologies are largely non intersecting.

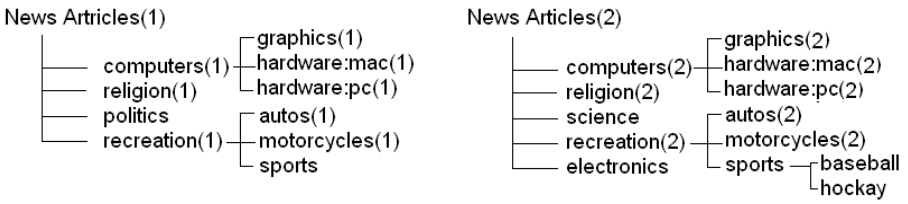

Fig. 1. Two news ontologies.

One could construct an ontology matching procedure based on a concept similarity measure by producing a set of N 1-to-M similarity assertions for an ontology with N and an ontology with M concepts. However, it is likely that such an initiative turns out to be rather costly, in spite of it being semantically unjustified, for in this case *structure* is not taken into account.

We suggest that the concept similarity measure should be applied recursively on the sets of concepts found on *corresponding* levels of the two ontologies, descending down the hierarchies. In a properly designed ontology, the classes on a single level (also referred to as *unilevel* classes) are internally homogeneous and externally heterogeneous. The search of potentially similar concepts is optimized by taking the concepts intra-ontology relations into account. The proposed recursive procedure stems from a simple rule: concept similarity is tested only for those cross-ontology concepts, whose parents have already been judged similar.

Considering the ontologies in Fig. 1, we start by mapping the set of concepts {*Computers(1), Religion(1), Politics(1), Recreation(1)*} against the set {*Computers(2), Religion(2), Science(2), Recreation(2), Electronics(2)*}. Let the mappings identified by the help of the similarity measure be {*Computers(1)* → *Computers(2)*}, {*Religion(1)* → *Religion(2)*}, and {*Recreation(1)* → *Recreation(2)*}. We proceed to map the sets of the children of each pair of concepts mapped in the first step (the children of the two *Computer*-classes and the children of the two *Recreation*-classes). The procedure stops when reaching the leaves of the trees.

[1] http://people.csail.mit.edu/jrennie/20Newsgroups/

```
procedure Map(Set_1, O_1, Set_2, O_2)
// Returns a set of mappings for the concepts of Set_1 and Set_2,
// and for their descendants in O_1 and O_2
begin
    if Set_1 = ∅ or Set_2 = ∅ then return ∅
    Σ = {} // Initialize the set of mappings
    for A ∈ Set_1 do // Find matches
        for B ∈ Set_2 do
            if sim(A, B) ≥ threshold then
                Σ := Σ + {(A, B)}
                break// ... to enforce injectivity of mapping
    if Σ ≠ ∅ then // Add mappings for descendants
        Mappings = Σ
        for (A, B) ∈ Σ do // Recursions
            Mappings = Mappings ∪ Map(children[A], children[B])
        return Mappings
    // Sigma is empty: Skip one level or more in O_2
    Σ_1=Map(Set_1, children[Set_2])
    if Σ_1 ≠ ∅ then  return Σ_1
    // None of the concepts in Set_1 could be mapped: skip this level of O_1
    return Map(children[Set_1], Set_2)
end

procedure Main(O_1, O_2)
begin
    // Find potential anchors and compute mappings for their descendants:
    for A in O_1 using breadth-first-search do
        for B in O_2 using breadth-first-search do
            if sim(A, B) ≥ threshold then
                return {(A, B)}∪ Map(children[A], O_1, children[B], O_2)
    return ∅
end
```

Algorithm 1: An algorithm for matching ontologies O_1 and O_2.

If our source ontologies are more complex than the ones considered above and are of different granularities, we have to ensure that at each step we are matching corresponding levels. To these ends, we suggest to set a threshold of the measured concept similarity: if the values found are under that threshold, the levels do not correspond and we should descend on the following level of O_2. A pseudo-code of the procedure is given in Algorithm 1, which also allows skipping of levels for the first ontology and additionally employs an anchoring technique. The procedure can be adapted for ontologies that contain other than strictly hierarchical relations in addition to a well-defined hierarchical backbone (see definition 1 and the comments thereafter) by matching the hierarchical backbones prior to the remaining concepts.

Table 1. Matching the news ontologies: sim_{proto}

Concept Names	Comp(2)	Religion(2)	Science(2)	Recr(2)	Electr(2)
Computers(1)	*0.924*	0.188	0.457	*0.514*	*0.6*
Religion(1)	0.175	*0.972*	0.154	0.201	0.191
Politics(1)	0.414	0.246	0.441	*0.522*	0.47
Recreation(1)	*0.539*	0.218	0.369	*0.843*	*0.586*

8 Experiments

In the following, we present experimental results obtained for the news ontologies in Fig. 1. We will start with the methods presented in Section 4, including the cosine similarity for class prototypes, $sim_{proto}(A, B)$ defined in equation 1, the Jaccard similarities $sim_{jaccard}$, defined below and the methods derived from clustering, $sim_{\min}$, $sim_{\max}$, and sim_{avg}. For these methods, we will only present the results for the top-level categories of the news ontologies shown in Figure 1. For the similarity measure based on the VC-dimension, we additionally evaluate the application of the recursive matching procedure (see Section 7).

In the test ontologies, leaf nodes were assigned about 500 documents on the related topic. Parent nodes were assigned the union of the documents assigned to their children plus some additional documents on their topic, in order to account for documents annotated directly by the parent concept. Each top-level has between 1900 and 2500 instances. Each instance is described by 329 features corresponding to a selection of the words occurring in the original news articles. Note that we reduced the initial number of terms substantially by removing stop words, applying stemming, and deleting high and low frequency words. The results are presented in the form of similarity matrices, where underlined entries mark the pairs of concepts that are supposed to be mapped onto each other, like *Computers(1)* and *Computers(2)*. Numbers in italics mark values above the threshold of 0.5. If Alg. 1 establishes a mapping for a pair of classes that are not supposed to be mapped, the mapping might be considered as incorrect.

8.1 Prototype Method

The prototype method 1 based on the Caiman idea simply consists in first computing the class (concept) means in the usual manner, and then applying the instance based similarity measure to it. Since all vectors have non-negative feature values, the similarity lies always between 0 and 1.

The similarities for the concepts in the two news ontologies can be found in Table 1. It can be determined that the prototype approach is able to detect similar pairs of concepts. However, it fails to properly detect dissimilar pairs, provided a natural threshold of 0.5 (see e.g. the pair *Politics(1)/Recreation(2)*). This makes it difficult to use this measure for the recursive matching procedure, which relies on being able to make such decisions. We assume that the relatively high similarity values for dissimilar concept pairs result from the averaging pro-

Table 2. Matching the news ontologies: $sim_{jaccard}$

Concept Names	Comp(2)	Religion(2)	Science(2)	Recr(2)	Electr(2)
Computers(1)	*0.597*	0.03	0.14	0.02	0.16
Religion(1)	0.0	*0.99*	0.0	0.0	0.0
Politics(1)	0.008	0.003	0.12	0.02	0.005
Recreation(1)	0.02	0.04	0.007	*0.86*	0.06

cess that is used for computing the means: compared to the more "extremal" vectors in each class, the means tend to be more similar.

8.2 Jaccard Similarity-Based Method

For completeness of our proof of concept, we tested the similarities of the first levels of the two news ontologies by the help of one of the most popular measures found in the extensional OM literature, the Jaccard similarity which is in the core of the GLUE matcher, given by

$$sim_{jaccard}(A, B) = \frac{P(A \cap B)}{P(A \cup B)}. \tag{8}$$

The quantities $P(A \cap B)$ and $P(A \cup B)$ were estimated by learning an SVM on the instances of O_1 and testing it on the instances of O_2 and vice versa, as explained in [4]. The concept similarities are found in Table 2. The results by using the *corrected* Jaccard coefficient, suggested by [9], were similar to the ones presented here. Below, these results will be compared with the results achieved with the proposed approaches.

8.3 Comparing the Sets of Support Vectors (SSV)

In the following, we present the results for comparing the sets of support vectors. The similarities for the sets are based on minimizing, maximizing, and averaging the similarities of pairs of instances for the two sets to be compared. The results can be found in Table 3.

The similarity values of $sim_{\min}$ are relatively low for all considered concept pairs and it also fails to correctly map *Recreation(1)* to *Recreation(2)*. $sim_{\max}$ shows the opposite effect and judges the similarity of all concept pairs as relatively high. For instance, the similarity value of *Recreation(2)* and *Politics(1)* is equal to 0.886 and thus much higher than the natural threshold of 0.5. sim_{avg} also attains relatively low values for all concept pairs, but in contrast to $sim_{\min}$ it can at least determine the most similar concepts for each concept correctly. Our conclusion is that all three measures present problems when being used in the matching procedure, since they require the user to choose a suitable threshold different from 0.5 for discriminating between similar and dissimilar concept pairs. Note that we also applied all three similarity measures to the full instance sets instead of the sets of support vectors. The findings were quite similar, but computation times were much higher.

Table 3. Matching the news-ontologies: $sim_{\min}$, $sim_{\max}$, and sim_{avg}

Concept Names	Comp(2)	Religion(2)	Science(2)	Recr(2)	Electr(2)
Computers(1)	0.00269	$1.97 \cdot 10^{-9}$	$3.03 \cdot 10^{-9}$	$6.99 \cdot 10^{-9}$	$4.76 \cdot 10^{-9}$
Religion(1)	$8.27 \cdot 10^{-9}$	0.0181	$1.29 \cdot 10^{-6}$	$4.46 \cdot 10^{-9}$	$4.44 \cdot 10^{-9}$
Politics(1)	$7.33 \cdot 10^{-9}$	$1.93 \cdot 10^{-9}$	$8.23 \cdot 10^{-9}$	$4.35 \cdot 10^{-9}$	$7.02 \cdot 10^{-9}$
Recreation(1)	$5.98 \cdot 10^{-9}$	$2.82 \cdot 10^{-9}$	$5.37 \cdot 10^{-9}$	$5.49 \cdot 10^{-9}$	$4.46 \cdot 10^{-9}$
Computers(1)	*0.889*	*0.503*	*0.713*	*0.892*	*0.917*
Religion(1)	*0.604*	*0.874*	*0.543*	*0.679*	0.629
Politics(1)	*0.867*	*0.568*	*0.839*	*0.886*	0.885
Recreation(1)	*0.921*	*0.536*	*0.746*	*1*	*0.919*
Computers(1)	0.0175	0.008	0.0122	0.0138	0.0148
Religion(1)	0.00808	0.0216	0.00848	0.00907	0.0086
Politics(1)	0.013	0.0109	0.0133	0.0147	0.0137
Recreation(1)	0.0164	0.00962	0.0133	0.0198	0.0165

Table 4. Matching the news ontologies by selecting variables with MI

Concept Names	Computers(2)	Religion(2)	Science(2)	Recr(2)	Electr(2)
Computers(1)	*0.548*	−0.405	0.119	0.146	0.464
Religion(1)	−0.242	*0.659*	−0.105	−0.201	−0.271
Politics(1)	−0.105	0.019	0.276	0.138	−0.015
Recreation(1)	0.318	−0.301	0.001	*0.528*	0.356

8.4 Similarity Based on VC Dimension (VC-VS)

We present an evaluation of the instance-based matching procedure suggested in Section 7 as well as of the finding that the VC dimension of SVMs can provide a criterion for selecting variables in a classification task. As a measure of similarity we have used Pearson's measure of correlation (wherefrom the negative numbers), given in (5). The measure indicates high similarity for positive values and low similarity for non-positive values. The results achieved by using the novel SVM-based variable selection technique (Table 5) proved to be better than those achieved by a standard point-wise mutual information-based criterion (Table 4). The results presented below come from using the former method.

After the root nodes of O_1 and O_2 have been matched, we proceeded to match the sets of their direct descendants. The results are shown in the upper similarity matrix on Table 5. Our similarity criterion identified successfully the three pairs of similar concepts (the Computers, the Religion, and the Recreation pairs). Following the matching procedure, as a second step we matched the sets of the descendants of the concepts that were found to be similar in the first step. The obtained similarity values for the children of the computer- and recreation-classes are shown in Table 6. Finally, in order to show the effect of not respecting the rule of hierarchical matching, we have matched the first level of O_2 with the descendants of the computer class in O_1. The obtained similarity values are

Table 5. Matching the news-ontologies by selecting variables with VC-SVM: first levels vs. mixed levels

Concept Names	Comp(2)	Religion(2)	Science(2)	Recr(2)	Elec(2)
Computers(1)	*0.78*	0.08	0.04	0.40	0.06
Religion(1)	0.07	*0.94*	0.02	0.10	0.01
Politics(1)	0.08	0.12	0.08	0.14	0.01
Recreation(1)	0.29	0.09	0.06	*0.60*	0.06
Graphics(1)	0.30	-0.01	-0.1	-0.01	-0.002
HW:PC(1)	0.18	-0.03	-0.01	-0.02	-0.03
HW:Mac(1)	0.03	-0.02	-0.01	-0.02	-0.09

Table 6. Similarities of the descendants of the computer- and the recreation-classes

Concept Names	Graphics(2)	HW-Mac(2)	HW-PC(2)	
Graphics(1)	*0.954*	-0.475	-0.219	
HW-Mac(1)	-0.266	*0.501*	-0.073	
HW-PC(1)	-0.577	0.304	*0.556*	
Concept Names	**Autos(2)**	**Motorcycles(2)**	**Baseball**	**Hockey**
Autos(1)	*0.978*	0.478	0.117	0.095
Motorcycles(1)	*0.560*	*0.989*	0.121	0.452
Sports	0.452	0.491	*0.754*	*0.698*

shown in the lower matrix in Table 5: although the potentially similar classes are accorded higher similarity values, the similarity coefficients are much lower than in the previous cases and much closer to the values for the dissimilar classes.

The similarity values are computed on the sets of input variables which, in case of text, correspond to actual words. Thus, the most important words that discriminate between a pair of similar concepts and the rest of the pairs of concepts can be readily made available in contrast to related methods (e.g. Caiman or Glue). For example, our selection procedure found out that among the most important tokens that characterize the concept *Computers* are *comp*, *chip*, *graphic*, *card*, *devic*, *file*. In contrast, the features with highest scores for *Religion* were *christ*, *church*, *faith*, *bibl*, *jesu*, for *Politics* – *polit*, *govern*, *legal*, *talk* and for *Recreation* – *motorcycl*, *auto*, *speed* and *engin*. This information is useful to verify the quality and coherence of the matching results.

9 Conclusion

The paper focuses on extension-grounded approaches to identify cross-ontology concept similarities by applying machine learning techniques for classification. Four similarity criteria have been tested on two source ontologies populated with textual instances: one based on comparing the support vectors learned per concept (SSV), one based on variable selection with VC-dimension (VC-VS), the Jaccard similarity used in the Glue tool and one, prototype method based on

the Caiman system. A matching procedure, using one of the proposed measures has been described and evaluated.

Our results showed that the VC-VS method outperforms the other considered measures, yielding a clearcut difference between the similarity values obtained for pairs of similar and pairs of dissimilar concepts. The method shows to respond properly to a natural similarity threshold of 0.5 on a 0-1 scale. Although the results achieved with the Jaccard similarity are competitive, the VC-VS approach makes a step further in terms of similarity verification, since the discriminant features (words) for each class are readily made available. The SSV technique, although inferior to VC-VS, tends to outperform the prototype method, provided an appropriate choice of similarity threshold from the user.

References

1. E. Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2004.
2. J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *JMLR*, 3:1229–1243, 2003.
3. C. Burges. A tutorial on support vector machines for pattern recognition. *DMKD*, 2:121–167, 1998.
4. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *WWW'02*, pages 662–673. ACM Press, 2002.
5. J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag Gmbh, 1 edition, 2007.
6. R. L. Goldstone and B. J. Rogosky. Using relations within conceptual systems to translate across conceptual systems. *Cognition*, 84(3):295 – 320, 2002.
7. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 3(1):1157–1182, 2003.
8. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.
9. A. Isaac, L. van der Meij, S. Schlobach, and S. Wang. An empirical study of instance-based ontology matching. *The Semantic Web*, pages 253–266, 2008.
10. T. Joachims. SVM light, http://svmlight.joachims.org, 2002.
11. M. S. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proceedings of the 14th FLAIRS Conf.*, pages 305–309. AAAI Press, 2001.
12. P. Mitra and G. Wiederhold. An ontology composition algebra. In S. Staab and R. Studer, editors, *Handbook on Ontologies*. Springer, 2004.
13. N. Noy and M. Musen. Anchor-prompt: Using non-local context for semantic matching. In *IJCAI'01*, pages 63–70, August 2001.
14. A. Rakotomamonjy. Variable selection using svm based criteria. *JMLR*, 3:1357–1370, 2003.
15. G. Stumme and A. Maedche. Fca-merge: Bottom-up merging of ontologies. In *IJCAI*, pages 225–230, 2001.
16. K. Todorov, P. Geibel, and K.-U. Kühnberger. Extensional ontology matching with variable selection for support vector machines. In *CISIS*, pages 962–968. IEEE Computer Society Press, 2010.
17. J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for support vector machines, 2001.