Structure- and Extension-Informed Taxonomy Alignment [*]

K. Selçuk Candan
Comp. Sci. and Eng.
Arizona State Univ.
candan@asu.edu

Mario Cataldi
Dip. of Informatica
Univ. of Torino
cataldi@di.unito.it

Maria Luisa Sapino
Dip. of Informatica
Univ. of Torino
mlsapino@di.unito.it

Claudio Schifanella
Dip. of Informatica
Univ. of Torino
schi@di.unito.it

ABSTRACT

Ontologies and concept taxonomies help software systems organize data more effectively for particular application domains. Ontologies also enable sharing and integration of data from different domains and data sources. However, ontologies from different domains are rarely identical; thus, there is need for techniques to find alignments between concepts in different ontologies and taxonomies. In this paper, we first note that alignment algorithms can be classified into two, *structurally-informed* and *extensionally-informed*. We then present a concept vector based scheme that captures structural information inherent in taxonomies to facilitate structure-based matching of concepts across taxonomies. We further note the structurally-informed concept vectors can further enable us to benefit from an available corpus of documents to implement *extensionally-informed* matching schemes with improved power of discriminating synonyms and homonyms of concepts.

1. INTRODUCTION

Ontologies and taxonomies are used in diverse areas of science, as well as in various standardization and information integration efforts, where the knowledge about the relationships of concepts is important. Given a concept taxonomy for a particular application domain, software systems can organize data more effectively. Thus, various knowledge-rich applications, including clustering algorithms, browsing support interfaces, and recommendation systems, perform more effectively when they are supported by domain describing ontologies, which help resolve ambiguities and provide context.

Ontologies also enable sharing and integration of information from different domains and data sources. However, ontologies from different sources are rarely identical; thus, there is need for techniques to find alignments between concepts in different ontologies and taxonomies.

[*]Authors are listed in the alphabetical order

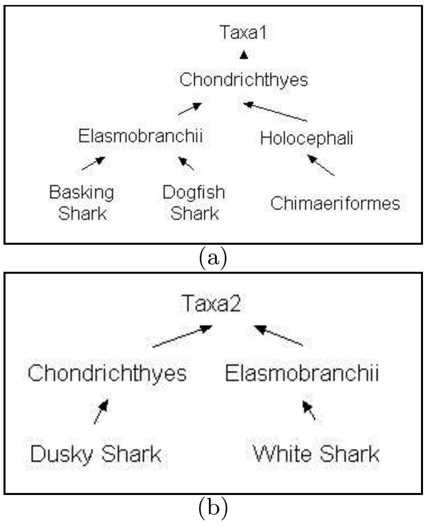


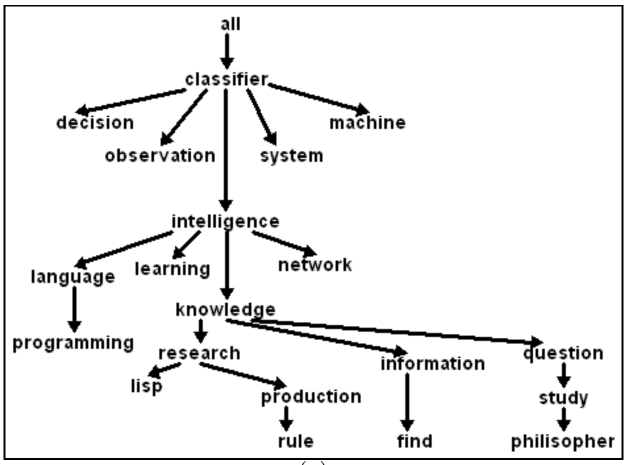
Figure 1: Two conflicting shark taxonomies

1.1 Motivating Application I: Archaeological Data Integration

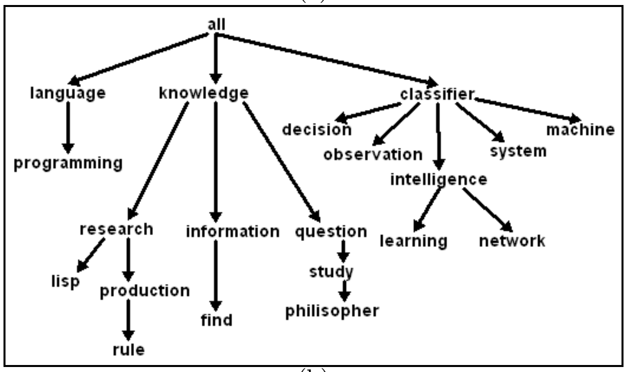
Through a joint effort of archaeologists and computer scientists, we are developing a framework of knowledge-based collaborative tools that will provide the foundation for a shared information infrastructure for archaeology and contribute substantially to a shared knowledge infrastructure of science[1]. Comprehensive, large-scale archaeological data are never collected by a single research team; data must be compiled from many projects. The incapacity to integrate data across projects cripples archaeologists' and other scientists' efforts to recognize phenomena operating on large spatio-temporal scales and to conduct crucial comparative studies [14, 30, 31]. A major challenge with integration of data is that the meaning of an archaeological observation is rarely self-evident. In archaeology, metadata that describe classification schemes, sampling and processing methods, and contexts of archaeological observations are essential to use datasets of any complexity.

Archaeological metadata are often hierarchical in nature; in particular, they include hierarchies of concepts (e.g. for biological and other taxonomies, see Figure 1, [30]). An important reality when integrating archaeological data is that entries (archaeological observations and interpretations) may

[1]NSF Grant, AOC: Archaeological Data Integration for the Study of Long-Term Human and Social Dynamics, 2007-2009



(a)



(b)

Figure 2: Two automatically-generated AI taxonomies obtained using the MAISON taxonomy extractor

often be “missing” or only partially specified. For example, one may not be able to associate a bone collected at a given site to the species and may use vague terms or references to a hierarchically higher concept in the biological taxonomy. Thus, researchers reach conflicting conclusions, not just because their primary data differ, but because they operationalize interpretive concepts differently. Nevertheless, in the context of a particular research question, archaeologists could identify reasonable means of addressing these inconsistencies [14]. Reconciling data and classification schemes entails developing novel alignment techniques to allow representation of integrated knowledge despite inherent mismatches.

1.2 Motivating Application II: Automatic Taxonomy Extraction and Integration for Effective Access to Digital Libraries

As part of the iCare project at Arizona State University, we are developing assistive and rehabilitative technologies for students who are blind and visually impaired. Our efforts include the MAISON project for developing a novel assistive system to help students and educators who are blind in accessing online digital records applicable for a teaching/learning task[2]. In particular, the content being registered into the National Science Digital Library (NSDL) is analyzed and adapted for effective search and navigation by individuals who are blind. This involves not only indexing and querying of text, but also understanding and merging *learning contexts* in which various educational material is offered through a *knowledge taxonomy extraction* process.

[2]NSF Grant “MAISON: Middleware for Accessible Information Spaces on NSDL”, 2008-2009

Figure 2 shows two similar, but different, AI taxonomies obtained using the MAISON taxonomy extractor[4] from different sources. As it can be seen here, the automatically generated taxonomies are imperfectly aligned: for example, in one of the taxonomies, the root has exactly one child, with label “`classifier`”, while in the other the root has three children, and “`classifier`” is one of them. This may be due to both imperfection in the taxonomy extraction process as well as the different contexts of the texts.

Thus management and integration of these taxonomies within MAISON, to help with precise searches, require mechanisms and models that are able to handle semantic conflicts and misalignments. In particular, they require techniques to recognize the nodes that (are likely to) denote the same concept in different taxonomies.

1.3 Contributions of this Paper

In this paper, we note that there is need for techniques to align taxonomies by taking into account both the taxonomy structure and the ”context” in which they will be used. More specifically, we observe that when taxonomies are used as information sources supporting navigation in a document space, the way users perceive the degree of matching between pairs of concepts is highly dependent on the domain being explored.

Consider for example the concepts ”German Shepherd” and ”White Shepherd”. These can be considered as different concepts as long as the scientific classification of dogs is concerned in a scientific domain where distinguished instances of these concepts can be identified. On the other hand, it might make sense to see them as matching concepts in a domain in which the distinction between the two species is not relevant, as it would be the case, for example, in a space of books which are novels having animals as their characters. If the taxonomy had to be used as a navigation tool for recommendation purposes, a broader classification of both ” Shepherds” as ”dogs” would probably be sufficient. This observation motivates our emphasis on the use of the domain data classification as input for the evaluation of the degree of matching between concepts.

We then argue that the advantages of the structure based and information based approaches can be combined in a *structure and extension informed* matching approach. Unlike the tree edit-distance based [32] or top-down/bottom-up [20, 16] schemes, our approach relies on both structural and extensional properties of the given concept taxonomies. It leverages the structural information coded by the hierarchical structure of the given taxonomies to infer matching relationships, that could not be directly detected through element-level matching techniques. The information about the extension of the concepts against a given database, contextualizes and grounds the taxonomy on the considered database and helps resolving conflicts/mismatches that might arise if only structural aspects are considered.

The paper is organized as follows. Section 2 shortly recalls the major ideas behind the existing structure and extension informed matching methods. Section 3 introduces an alternative structure based approach that uses concept-vectors for matching concepts between taxonomies. Section 4 combines the purely structural approach with extension based matching evaluation, thus resulting in a method that is structure-and extension-informed at the same time. Section 5 evaluates the proposed method and shows its effectiveness on our test cases. Section 6 concludes the paper and presents our future work plan.

2. RELATED WORK

The problem of matching context-describing taxonomies has been investigated in various application areas, especially in scientific, business, and web data integration [27, 8, 20, 18, 5, 22, 16, 24, 3]. Different matching techniques focus on different dimensions of the problem, including whether data instances are used for schema matching, whether linguistic information and other auxiliary information are available, and whether the match is performed for individual elements (such as attributes) or for complex structures[27].

Cupid [16] is a schema-based approach that implements a sequential composition of different matchers. It consists of a first phase based on a linguistic matcher and a second phase based on a structural matching technique. The linguistic matcher calculates similarity coefficients between schema label nodes, while the structural matcher computes similarity values which measure the similarity between contexts in which elementary schema elements occur. A final phase aggregates these results by means of a weighted sum and compares them with a given threshold in order to generate the alignment. This algorithm operates only with trees: other schemas can be handled through a translation process. [20] uses schema graphs for matching; matching is performed node by node starting at the top; thus this approach presumes a high degree of similarity (i.e., low structural difference) between the taxonomies. Onion [22], the successor of SKAT [21], is a schema-based system that leverages logic rules to discover match and mismatch relationships between multiple ontologies, internally represented as labeled graphs. The matching algorithm proposes a sequential (and semi-automatic) approach that first performs a linguistic match and then applies a structure-based matching. The latter is based on the result of the first step and tries to match only the unmatched terms; it is based on a structural isomorphism detection technique between the subgraphs of the ontologies. [3] and DIKE [25] use the distance of the nodes in the schemas to compute the mappings; while computing the similarity of a given pair of objects, other objects that are closely related to both count more heavily than those that are reachable only via long paths of relationships. Glue [9], the successor of LSD [6], is an instance-based semi-automatic system that uses machine-learning techniques to discover one-to-one mappings between two taxonomies. It is based on the joint distributions that are used for any similarity measures. This approach can be divided in three steps. First, a multi-strategy learning approach allows to compute the joint distribution of classes that are used in the second step to produce a similarity matrix. The latter is used in the final phase by a relaxation labeling technique in order to filter only the best matches contained in the similarity matrix. Differently from Glue, FCA-merge [1] takes as input two ontologies that share the same set of instances and returns a new ontology. It uses formal concept analysis techniques, through a three steps process: instance extraction, concept lattice computation, and (interactive) generation of the final new ontology. Clio [18, 19] is a mixed schema-based and instance-based system that proposes a declarative approach to schema mapping between XML and relational schemas. After the first phase in which input schemas are translated into an internal representation, the system sequentially combines an instance-based attribute classification (by using a Bayes classifier) with a string matching between elements' names. These n-to-m value correspondences can be also entered by the user through a graphical user interface. Then, Clio produces a set of logical mappings with formal semantics, also supporting mappings composition. [8] provides a more detailed classification of matching techniques, based on other features including different similarity measures, matching strategies (such as name similarity or class similarity), and degrees of user involvement. [11] proposes an algorithm for ontology matching which combines standard string distance metrics with structural similarities computed using an iterative algorithm.

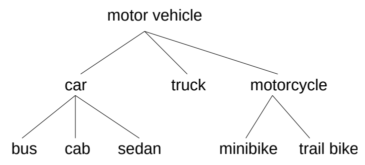


Figure 3: An IS-A hierarchy: "bus" and "cab" are more closely related than "bus" and "minibike".

Despite such advances in structural mapping technologies, alignments across data sources are rarely perfect. In particular, imperfection can be due to homonyms (i.e., nodes with identical concept-names, but possibly different semantics, in the given taxonomy hierarchies) and synonyms (concepts with different names but same semantics). While structural-matching techniques help finding node-to-node alignments, they fall short when such scenarios arise.

3. STRUCTURE INFORMED ALIGNMENT USING CONCEPT VECTORS

Many knowledge-driven applications (such as text classification, word sense disambiguation, and data mapping) require mining of semantic similarity/ dissimilarity values between concepts in a given domain. Therefore, the study of semantic relationships between words in a language has a long history in psychological theory, natural language processing, and knowledge management. There are various general purpose efforts, such as WordNet [17] and FrameNet [2], to model the lexical knowledge underlying a language in the form of a hierarchical taxonomy, where the structure of the graph represents the knowledge about the relatedness of the words. Intuitively, highly related words are grouped together and the path between two different concept-nodes in the hierarchy reflects how these are related in the real-world.

If, for example, we consider the WordNet segment presented in Figure 3, we can intuitively see that the two concepts, "*bus*" and "*cab*", are more closely related to each other than concepts, "*bus*" and "*minibike*".

3.1 Concept Similarity Computation and Concept-Vectors

In the last decade various measures for estimating the semantic similarity of keywords in a given taxonomy have been proposed. These measures can be roughly categorized into

Table 1: Concept vectors for the nodes in Figure 3

	motor vehicle	car	truck	motorcycle	bus	cab	sedan	minibike	trail bike
$cv_1(motorvehicle)$	0.450	0.151	0.152	0.152	0.0176	0.018	0.018	0.021	0.021
$cv_2(car)$	0.052	0.4697	0.006	0.006	0.156	0.156	0.156	0.0003	0.0003
$cv_3(truck)$	0.100	0.012	0.873	0.012	0.0006	0.0006	0.0006	0.0007	0.0007
$cv_4(motorcycle)$	0.057	0.007	0.007	0.520	0.0003	0.0003	0.0003	0.204	0.204
$cv_5(bus)$	0.004	0.100	0.0002	0.0002	0.872	0.012	0.012	0	0
$cv_6(cab)$	0.004	0.100	0.0002	0.0002	0.012	0.872	0.012	0	0
$cv_7(sedan)$	0.004	0.100	0.0002	0.0002	0.012	0.012	0.872	0	0
$cv_8(minibike)$	0.006	0.0003	0.0003	0.165	0	0	0	0.806	0.023
$cv_8(trailbike)$	0.006	0.0003	0.0003	0.165	0	0	0	0.023	0.806

structure-based (or, edge-based) methods and *information-based* methods. In structure-based methods, the semantic similarity between two words is measured by the shortest distance between them [26] or the sum of the edge weights along this shortest path [29]. Information-based methods leverage available corpora to extract additional information, such as keyword frequency, to achieve better similarity evaluation than those approaches that rely only on the structural analysis of a hierarchy. For example, [28] estimates the similarity between two concepts using the information content (i.e., negative logarithm of the probability of encountering an instance of the concept in the corpus) of the concepts subsuming them. When using [28], the similarity between "*bus*" and "*minibike*" in Figure 3 would be determined by the information content of the node "*motor vehicle*", which subsumes both words in the hierarchy.

Recently, [13] showed that there is an alternative approach to leverage structural knowledge inherent in taxonomies to compute the semantic similarity between nodes. The proposed algorithm, *concept propagation/concept vectors* (CP/CV), significantly improves semantic similarity measurements on a given taxonomy, when compared with other structure-based methods and gives as good or better results than information-based methods. Given a taxonomy, CP/CV assigns a *concept-vector* to each concept node in the taxonomy, such that the vector encodes the *structural* relationship between this node and all the other nodes in the hierarchy. The concept vectors are obtained by propagating concepts on the taxonomy graph according to their *semantic* contributions (dictated by the structure of the taxonomy). In CP/CV, given a hierarchy, $\mathcal{H}(N, E)$ with m concept-nodes, each node is mapped onto a concept-space with m concept-dimensions. Initially, each concept-node is mapped into the concept-space along the dimension corresponding to itself. Then, for each pair, n_i, n_j, of neighboring concepts, CP/CV computes two values, $G^{str}_{n_i,n_j}$ and $G^{cs}_{n_i,n_j}$. G^{str} measures the *degree of generality* between two entities computed using the taxonomical structure. G^{cs}, on the other hand, computes a corresponding value using the concept space. Since these hierarchical and multi-dimensional representations are different views over the same semantics, CP/CV computes concept propagation parameters that preserve the equality

$$G^{cs}_{n_i,n_j} = G^{str}_{n_i,n_j}$$

after propagation. This process is iterated until all nodes are informed of all the others. Once the process is computed, since all the concepts are mapped into the same vector space, semantic similarities of the concepts are computed using the cosine similarity of the concept vectors. Measuring the semantic similarities across different taxonomies, using the cosine measure of concept-vectors directly, would not be possible as the concept-spaces corresponding to different taxonomies are also different. As shown in [13], this leads to a very precise similarity measure of concepts within a taxonomy.

EXAMPLE 3.1. *Consider the taxonomy fragment containing the nine nodes presented in Figure 3. Each concept is represented by a 9-dimensional vector. Vector's elements are associated to the taxonomy nodes, considered in breadth first order. In particular, the root is represented by the vector*

$$\langle 0.450, 0.151, 0.152, 0.152, 0.0176, 0.018, 0.018, 0.021, 0.021\rangle,$$

in which the first component - the one associated to the tag "motor vehicle", dominates over the others that contribute to the definition of the concept. The second, third and fourth components reflect the weight of "car", "truck" and "motorcycle" respectively in the semantic characterization of "motor vehicle", while the remaining components represent the weights of the three descendants of "car" and of the two descendants of "motorcycle". Similarly, the concept vectors for all nodes are as shown in Table 1.

3.2 Matching Concepts across Taxonomies through Concept Vectors

In this paper, we build on the idea of concept-vectors [3] for matching concepts across taxonomies. The concept-vectors obtained by processing the hierarchical structure of the given taxonomies allow us to infer new matching relationships, not directly detectable through element-level matching techniques. A node in a given hierarchy clusters all its descendant nodes and essentially acts as a *context* for the descendant nodes (Figure 4(a)). Similarly, descendants of a given node may also act as a context for the node (Figure 4(b)), differentiating the node from others that are similarly labeled. Consequently, one way to differentiate nodes from each other is to infer the contexts imposed on them by their neighbors, ancestors, and descendants in the given hierarchy, enrich (or annotate) the nodes using vectors representing these contexts, and compare these context-vectors along with the label of the node. While CP/CV vectors are able to capture the descendant supplied context as well as the ancestor supplied context, most other structural schemes rely only on ancestor supplied context in alignments.

[3]These *concept vectors* are different from *concept vectors* introduced in [11] and used for the purposes of alignment. In [11], the concept vector of a node in taxonomy T_1 represents the degree of matching of this node to all nodes in another taxonomy T_2. The concept vectors introduced in [13] and also used in this paper, on the other hand, represent the relationships of the nodes in a single taxonomy among themselves. Consequently, the concept vectors for each taxonomy are created independently and thus can be reused for matching a given taxonomy to many others.

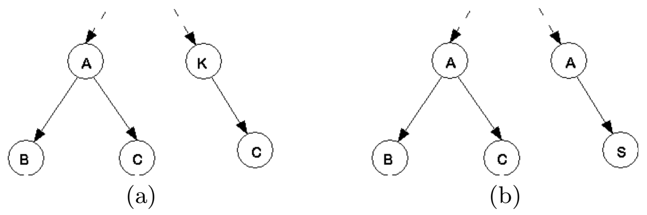


Figure 4: (a) Ancestor-supplied context differentiating the two "C"s and (b) descendant-supplied context differentiating the two "A"s

Let $\mathcal{H}_1(N_1, E_1)$ and $\mathcal{H}_2(N_2, E_2)$ be two taxonomies. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two concept spaces corresponding to $\mathcal{H}_1$ and $\mathcal{H}_2$. Note that $\mathcal{S}_1$ and $\mathcal{S}_2$ have $|N_1|$ and $|N_2|$ dimensions, respectively. Let also $\mathcal{C}_1$ and $\mathcal{C}_2$ be the sets of concept-vectors corresponding to the individual concepts in both hierarchies ($|\mathcal{C}_1| = |N_1|$ and $|\mathcal{C}_2| = |N_2|$).

A particular challenge that arises when matching taxonomies using two different concept-spaces is that, before the concept-vector similarities can be computed, one needs to match and align the dimensions of the concept-spaces, themselves. Depending on the domain application, and on the expected use of the taxonomies (i.e., the expected classification method) different approaches can be adopted. If the concepts associated to the nodes of the taxonomy are denoted by single word labels, and the expected classification is string match based (as it is the case, for example, if the database objects are classified based on the occurrence of label keyword in their description, or in some attribute/field), the space alignment needs to rely on syntactical evidences; i.e., matches between the "names" of the concepts. For example "*german-shepherd*" and "*white-shepherd*" are similar concept names. Unfortunately, syntactic similarities may not always correspond to semantic alignments (e.g., "*german-shepherd*" and "*shepherd*" are also syntactically similar, but while the first one identifies a class of dogs, the second one is likely to identify a class of human beings). While such syntactic matches across concept names cannot be used alone to measure semantic similarity, they can be leveraged as starting point while aligning the dimensions of the two spaces.

We note that (especially when fuzzy syntactic similarities are considered for aligning dimensions) the dimensions of the concept-space are potentially pairwise correlated. This is likely to introduce some bias in the measure of the degree of matching among nodes. These biases can be taken into account by using generalized measures, such as the following generalized *dot product*

$$dotprod\_gen(\vec{cv}_1, \vec{cv}_2) = \vec{cv}_1 \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,m} \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_{n,1} & \alpha_{n,2} & \cdots & \alpha_{n,m} \end{bmatrix} \vec{cv}_2^{\,T},$$

where $\alpha_{i,j}$ accounts for the lexical similarity between the i^{th} concept in the first taxonomy and the j^{th} concept in the second.

Given a concept $n_i \in N_1$, we compute a fuzzy match in the aligned space between n_i and all $n_j \in N_2$. This provides a list of candidates in taxonomy, $\mathcal{N}_2$, potentially matching $n_i \in N_1$, in decreasing order of likelihood. As an example, reconsider the two fragments in Figure 1. Here, after concept propagation, the concept vector associated to "`Chondrichthyes`" in the first fragment will capture the existence of its relationship with ``Helasmobranchii'' and ``Holocephali''. When "`Chondrichthyes`" is matched against the concepts in the second taxonomy, this will most likely identify two candidate matches for ``Chondrichthyes''; ``Helasmobranchii'' and ``Holocephali''.

4. EXTENSION-INFORMED ALIGNMENT

While concept vectors help us capture and leverage the structural information embedded in taxonomies for matching their concept nodes, in this paper we note that they can also enable us benefit from an available corpus of documents to improve matching performance. The main idea behind the *extensionally-informed* matching is that

> "if two concepts in two different taxonomies are to be considered *similar* in the considered context, then they will *relate to similar documents* in the corpus; in contrast, if two concepts are to be considered *different*, then the documents that relate to these concepts will also be *different*."

In line with the common notion in literature, we call the documents that a concept relates to as its *extension*.

One trivial way of identifying *extensions* of concepts is to find documents that contain the corresponding concept name as a keyword. This, however, cannot be used for taxonomy alignment:

- synonyms in two different taxonomies will have identical extensions, and
- homonyms in the taxonomies will have (most likely) very different extensions.

Instead, we need a mechanism that identifies *extensions* of concept nodes without relying only on their concept names.

The concept vectors associated to the concept nodes provide a convenient way to identify extensions. In particular, we rely on a classifier module which takes as input the set $CV = \{cv_1, \ldots, cv_m\}$ of the concept vectors representing the taxonomy, and the set $V = \{v_1, \ldots, v_n\}$ of vectors representing the documents to be classified. Keyword vectors are defined in the space of the entire set of document keywords; each dimension corresponds to a keyword, and the weights in the vector represent the relevance of the corresponding keyword value in the document represented by the vector. The goal of the classification component is to *associate the database documents to their best representative concepts in the taxonomy.* We capture this notion of representativeness through the notion of the similarity among the taxonomy and document vectors representing taxonomy concepts and documents, respectively. Semantic similarities (at the basis of the classification process) between the concepts and the objects being classified are computed by

- unifying the vector spaces of the concepts and of the objects. The unification of the spaces consists in unioning the dimensions in the given ones, and representing every vector in the new extended space by setting to 0 the values corresponding to those dimensions that were not appearing in the original vector space, while keeping all the other components unchanged.
- using the dot product similarity of the vectors.

In the following discussion we will always assume to deal with vectors sharing the same space.

4.1 Document-to-Concept Classification

For every document in the corpus, the classification identifies the taxonomy concepts that best match with it. Each document is considered as belonging to the extensions of those concepts whose similarities with it are above an adaptively computed critical point. The classification steps are the following:

1. For each database document
   (a) calculate the vector v_j, where each element contains the augmented normalized term frequency [10] of a document keyword.
   (b) compute its similarity wrt. all the concept vectors describing the given taxonomy.

$$sim(cv_i, v_j) = \Sigma_{k=1}^{u} cv_{ik} \times v_{jk}$$

   (c) sort the concepts vectors in decreasing order of similarity wrt. v_j;
   (d) choose the cut-off point to identify the concepts which can be considered *sufficiently similar* to justify the classification of the object under them. Our method adaptively computes a cut-off as follows: It
      i. first ranks the concepts in descending order of match to v_j, as previously calculated in (c).
      ii. computes the *maximum drop* in match and identifies the corresponding drop point.
      iii. computes the *average drop* (between consecutive entities) for all those nodes that are ranked before the identified maximum drop point.
      iv. the first drop which is higher than the computed average drop is called the *critical drop*. We return concepts ranked better than the point of critical drop as candidate matches.

At the end of this phase, all documents in the corpus are associated to the extension of at least one concept, provided that there is at least one concept in the taxonomy whose similarity with the document is greater than zero. Notice that the extensions of different concepts are not disjoint, since the same object can be assigned to multiple (similar) concept vectors. The number of concepts associated to a given document depends on the corresponding adaptive threshold value computed by the classification algorithm.

4.2 Extension-based Matching

The extensions computed for nodes are then used for supporting extensionally-informed matching between pairs of nodes. In particular, we consider as highly matching those concepts whose extensions are very similar. Thus, the definition of the matching reduces to the notion of similarity of the extensions. The extension based matching is based on the notion of *classification*: given a document d and a concept c, it considers the document d as either belonging or not belonging to the extension of c, without any other information about the degree of membership.

Definition 1. **(Extension based Matching)** Let c_i and c_j two concepts, and E_{c_i} and E_{c_j} their corresponding extensions. Their degree of matching is computed as

$$CEM(c_i, c_j) = \frac{|E_{c_i} \cap E_{c_j}|}{|E_{c_i} \cup E_{c_j}|}.$$

This can be seen as a special case of the Jaccard Similarity [12, 15], which given two sets A and B, and the probability distribution $P(X)$ (probability of a random instance to be in the set X), defines the similarity between A and B as $Jaccard\_sim(A, B) = \frac{P(A \cap B)}{P(A \cup B)}$. Essentially, this extension-based matching technique leverages both structural knowledge embedded in the concept vectors as well as the context provided by the corpus of documents. In the next section, we will experimentally evaluate the impact of extensional knowledge on the alignment process.

5. EXPERIMENTAL EVALUATION

To run our evaluation experiments, we considered a taxonomy extracted from the *DMOZ* scientific concepts categorization accessible at the link *http://www.dmoz.org/Science/*. The considered taxonomy has 73 nodes, depth 4, and different branching factors in its internal nodes (the average value is 5.14). To test the extension based approach, we have classified 17420 article abstracts describing NSF awards for basic research [4].

To evaluate the effectiveness of the proposed matching strategy in the presence of different conditions, we created a second taxonomy to be matched against, by introducing controlled distortions to the first one. For each of the following conditions we applied distortions of the order of 20% and 40% of the nodes.

- *Synonyms :* we randomly picked a percentage of nodes, and *relabeled* their concept names with other keywords (without affecting the structure of the taxonomy). Note that the new strings are actually random –i.e., not "synonyms" in English language- thus they do not actually occur in the documents. This constitutes a worst case situation for extension based algorithms.

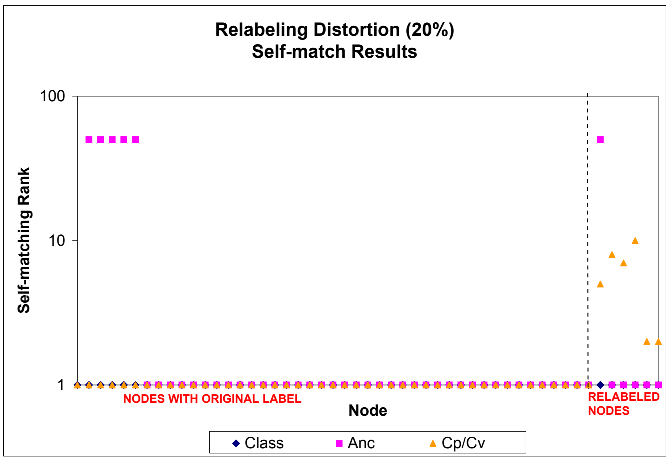
- *Homonyms:* we randomly picked a percentage of nodes, and for each of them introduced a replica in randomly selected positions of the other taxonomy. The replica has the same concept name, but is contextualized in a different position in the structure of the taxonomy, thus it denotes a different concept (i.e., homonym).

We then compared *extension-based (*`Class`*)* matching to two pure structural matching techniques, `CP/CV`, which considers both ancestor and descendant supplied contexts, and *closest-common-ancestor distance (*`Anc`*)*, which considers only ancestor supplied context.

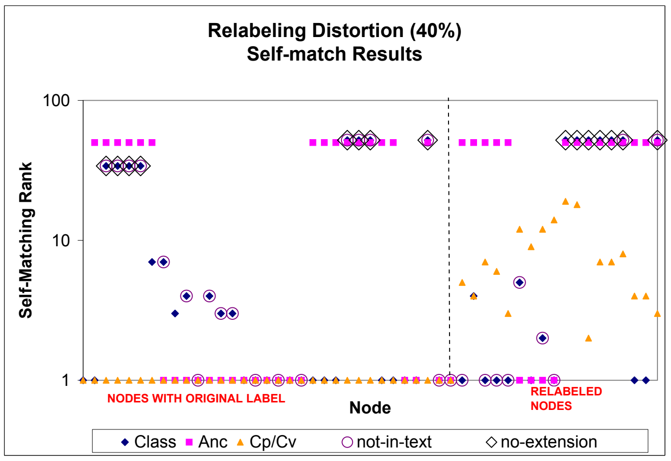
Figures 6(a) and (b) show the *synonym matching* results. In this figure, the X axis denotes the nodes and Y axis denotes the rank at which the corresponding node is found in the distorted taxonomy. Note that if the alignment algorithms work perfectly, then relabeling does not have any impact and all nodes are found at rank 1.

- For the 20% distortion case (Figure 5(a), the portion relabeled is on the right), we see that `Class` returns perfect match (100% of perfect matches). `Anc` makes some mistakes when a node near the root is relabeled (86.3% of perfect matches). `CP/CV` on the other hand works very well for non-relabeled nodes, but (since it

[4]http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html



(a)



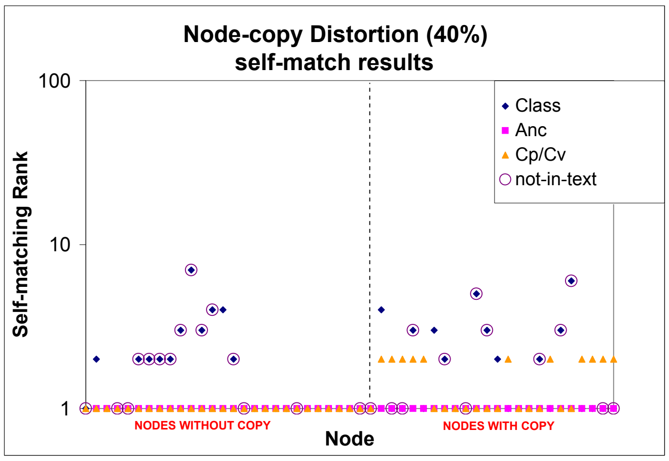
(b)

Figure 5: Matching results under concept relabeling.(rank=1 indicates perfect match)

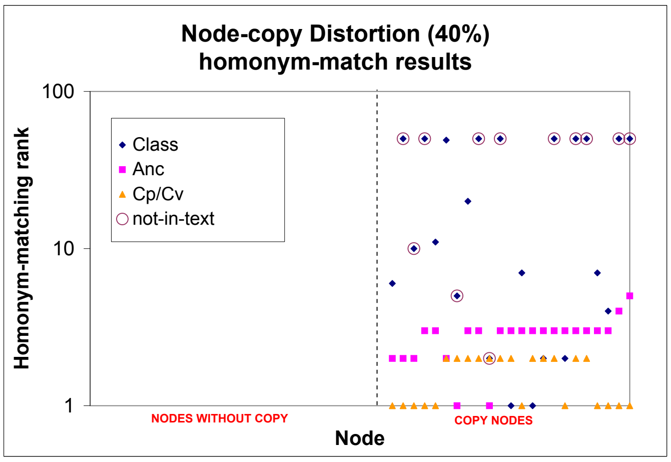
relies on the concept labels to some degree) it performs imperfectly for re-labeled nodes (88.3% of perfect matches). In short, while it is based on the CP/CV vectors, `Class` improved the results thanks to available extensions.

- When 40% of the nodes are arbitrarily relabeled (Figure 5(b), the portion relabeled is on the right), this has a significant impact on the performance: `Anc` makes significant errors across the board (42.2% of perfect matches). `CP/CV`, once again, works very well for non-relabeled nodes and performs imperfectly (but not very bad) for re-labeled nodes (totally 64.8% of perfect matches). The performance of `Class` in this heavily distorted situation depends on whether the terms support classification. Except for the cases in which the random labels inserted into the taxonomy prevents classification of the corresponding node[5], `Class` performs very well (83.4% of perfect matches).

Figures 5(a) and (b) show the *matching* results in the presence of multiple concepts with identical labels (we only show heavy, 40%, distortion). Figure 5(a) presents the matching



(a)



(b)

Figure 6: Matching results under homonyms. For (a) rank=1 indicated perfect match. for (b) i.e., homonyms, it is better to have match rank $\gg$ 1.

ranks for corresponding nodes, while Figure 5(b) shows the matching rank for the homonyms. Note that, for the former case, the closer to 1 the rank, the better the results are; whereas, for the homonym case, the further from 1 the ranks, the more discriminating the algorithm is.

- For self-matching cases (Figure 6(a)), since the node-copy distortions do not affect any internal nodes, `Anc` works perfectly (100% of perfect matches). `CP/CV`, which gets structural context also from descendants introduces some errors (rank 2 instead of 1) but, generally works well (78.5% of perfect matches). `Class`, on the other hand, works well unless the concept does not occur in the corpus at all[6] (91.2% of perfect matches): randomly copied concepts distort structural contexts in the destination positions and, thus, prevent other concepts from leveraging the corpus using their own concept-vectors.
- For homonym-matching cases (Figure 6(b), we see that `Class` works the best (puts the homonym furthest away from rank=1), whether the concept occurs in the corpus or not. In both cases, while the original concept

[5]These cases are marked with ◇ and ○

[6]These cases are marked with ○

is able to leverage the context provided by its neighborhood to classify documents, the arbitrarily picked context of the copy does not classify similar documents and homonyms are strongly identified.

5.1 Summary

It is important to note that the experimental evaluation shows that neither approach can be considered as better than the others under all situations. This is because, different methods leverage different hints. In particular, extension based methods perform better than purely structure based ones in those concepts for which a "significant" extension exists in the corpus.

6. CONCLUSIONS

In this paper, we have shown that *concept vector*s can be used for both capturing the inherent structure of taxonomies for implementing *structural matching* algorithms and for identifying documents relating to the individual concepts of the taxonomies to implement *extensional matching* schemes. Experimental results showed that when combined, structural and extensional techniques provide superior handling of synonym and homonyms.

To leverage the good performances of both methods, we are working on a combined matching strategy, that gives higher weight to the extension based approach whenever populated concept extensions are available, while relying on our structure based approach when measuring the degree of matching of concepts whose extension in the considered domain documents is empty.

7. REFERENCES

[1] G. Stumme and A. Maedche. Fca-merge: Bottom-up merging of ontologies. In *IJCAI*, pages 225–234, 2001.

[2] C.F. Baker, C.J. Fillmore, and J.B. Lowe. The Berkeley FrameNet Project. In *COLING*, 1998.

[3] K. S. Candan,J. W. Kim, Huan Liu, R. Suvarna Discovering mappings in hierarchical data from multiple sources using the inherent structure. *J. of KAIS*, 2006.

[4] L. Di Caro Bootstrapping di tassonomie basato su tecniche di analisi statistica del testo. Master thesis, University of Torino, September 2007 (In Italian).

[5] A. Doan, P. Domingos, A. Y. Levy. Learning source description for data integration. *WebDB*, 81-86, 2000.

[6] A.Doan, P. Domingos, A. Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. SIGMOD '01, 2001, pages 509–520.

[7] Eckart, C., Young, G. The approximation of one matrix by another of lower rank. Psychometrika, I, 211-218, 1936.

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

[9] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Ontology matching: A machine learning approach. *Handbook on Ontologies*, pages 385–404. Springer, 2004.

[10] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[11] A. Hess. An Iterative Algorithm for Ontology Mapping Capable of Using Training Data. *ESWC* 2006.

[12] Jaccard, The distribution of the flora of the alpine zone. New Phytologist. Vol 11,pahes 37-50, 1912.

[13] Jong Wook Kim, K. Seluk Candan CP-CV: Concept Similarity Mining without Frequency Information from Domain Describing Taxonomies, CIKM 2006.

[14] Kintigh, Keith W. (Ed.) The Promise and Challenge of Archaeological Data Integration. In *American Antiquity* 71(3): 567-578, 2006

[15] M. Lalmas, Information Retrieval: Uncertainty and Logics: Advanced Models for the Representation and Retrieval of Information. Cornelis Joost Van Rijsbergen editor, Kluwer Academic Publishers, 1998.

[16] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. *VLDB*, 2001.

[17] G.A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. Introduction to WordNet: An on-line lexical database. Int. J. of Lexicography, 3(4), 1990.

[18] R. Miller, L. Haas, M. Hernandez. Schema mapping as query discovery. *VLDB*, 2000.

[19] R. J. Miller, M. A. Hernandez, L.M. Haas, L. Yan, C.T. H. Ho, R. Fagin and L. Popa, The Clio Project: Managing Heterogeneity, ACM SIGMOD Record, 30(1), 2001.

[20] T. Milo, S. Zohar. Using schema matching to simplify heterogeneous data translation. *VLDB*, 1998.

[21] P.Mitra, G. Wiederhold, J.Jannink, Semi-automatic integration of knowledge sources. Proc. of the 2nd Int. Conf. On Information FUSION, 1999.

[22] P.Mitra, G.Wiederhold, M.Kersten. A graph oriented model for articulation of ontology interdependencies. *EDBT*, 2000.

[23] E. Myers. An O(ND) Difference Algorithms and its Variations. Algorithmica, 1(2), pp. 251-266, 1986.

[24] L. Palopoli, D. Sacca, D. Ursino. An automatic technique for detecting type conflicts in database schemes. *CIKM98*

[25] L. Palopoli, G. Terracina and D. Ursino DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases, Softw. Pract. Exper Journal,33(9), 2003.

[26] R. Rada, et al. Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man and Cybernetics, 19(1), 1989.

[27] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4), 2001

[28] P. Resnik. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. JAIR, Vol.11, 1999.

[29] R. Richardson and A.F. Smeaton. Using WordNet in an Knowledge-Based Approach to Information Retrieval. Working paper CA-1294, Dublin City Univ., Dublin, 1994.

[30] Y. Qi, K.S. Candan, and M.L. Sapino. Feedback-based InConsistency Resolution and Query Processing on Misaligned Data Sources, SIGMOD 2007

[31] Y. Qi, K.S. Candan, M.L. Sapino, and K.W. Kintigh. Integrating and Querying Taxonomies with QUEST in the Presence of Conflicts. SIGMOD 2007.

[32] K. Zhang,J. Wang,D. Shasha. On the Editing Distance between Undirected Acyclic Graphs, IJCS 1996.