

Automatic Extraction of Structurally Coherent Mini-Taxonomies

Khalid Saleem¹ and Zohra Bellahsene¹

LIRMM - UMR 5506 CNRS University Montpellier 2,
161 Rue Ada, F-34392 Montpellier
{saleem, bella}@lirmm.fr

Abstract. In this paper we demonstrate an automatic approach for emergent semantics modeling of ontologies. We follow the collaborative ontology construction method without the direct interaction of domain users, engineers or developers. A very important characteristic of an ontology is its hierarchical structure of concepts. Semantic web is heavily dependent on the XML paradigm, which inherently follows the hierarchical structure. We consider large sets of domain specific schemas as trees and apply frequent sub-tree mining for extracting common hierarchical patterns. Our experiments show that these hierarchical patterns are good enough to represent and describe the concepts of the domain ontology. The technique further demonstrates the construction of the taxonomy of domain ontology. In this regard we consider the largest frequent tree or a tree created by merging the set of largest frequent sub-trees as the taxonomy. We argue in favour of the trustability for such a taxonomy and related concepts, since it has been extracted from the schemas being used with in the specified domain.

Keywords: Ontology Learning, Mini-taxonomies, Collaborative Ontology Construction, Tree Mining, Large scale

1 Introduction

Over the years technology has made this world a web of digital information, where digital systems are appearing at an exponential rate. At individual level, personal or professional, or organisational level, there exists an unending list of digital devices cooperating together to solve problems. Every day a new gadget hits the market, creating a ripple-effect in its surrounding operating environment. Thus giving rise to new innovations in the field around it. For us, the database people, it is like emergence of new form of data or information, which has to be utilised in the most efficient and effective manner. The ability to exchange and use of data/information between different devices (physical or logical), is the basic activity in any type of system, usually referred to as data interoperability [17]. Thus the domain of data interoperability has also evolved with emergence of new devices and systems.

Today the central platform being utilised to share the digital information is the World Wide Web, which is evolving from an unstructured data presenter to a more semantically structured entity termed as Semantic Web. Different types of information sharing (P2P), processing (multi-agent) and delivering (web search engines) systems have been developed to harness the power of Web. Semantic Web provides a platform where machines can move one step further and understand the structures of data and the contextual meaning of the data. One of the most promising technique in this regard has been the ontology. Ontology is considered to be a complete semantic construct applicable to every field of computing. In short it is becoming the backbone of Semantic Web [3]. Its utilisation has been demonstrated from simple schema matching for data integration [16] to large scale complex web services management [2] ¹.

There have been several works in regard of ontology development, manual and semi-automatic. Ontologies have been build from scratch and from already available data content, in the form of text [4], web [7, 14, 21], tables [20], relational schemas [13], XML schemas and documents [10]. In all these works the ontological constructs have been the same; terms, concepts, concept hierarchies, relations and rules or logic. These features of an ontology have been described in detail by Paul Buitelaar et al. [4] as an ontology learning layer cake. These ontology features have a direct relation to the layered approach of Semantic Web [1].

In this paper we propose a novel approach for finding mini-taxonomies representing certain domain concepts using tree mining techniques. The approach is further extended to build a base taxonomy for the domain ontology. We emphasize upon the automatic aspect of our approach.

Tree mining techniques extract similar sub-tree patterns from a large set of trees and predict possible extensions of these patterns. A pattern starts with one node and is incrementally augmented. There are different techniques [6] which mine rooted, labeled, embedded or induced, ordered or unordered subtrees. The function of tree mining is to find sub-tree patterns that are frequent in the given set of trees. We utilise this aspect of tree mining for computing the mini-taxonomies for concepts and base taxonomy for the domain ontology, from a given set of schemas, input as trees. Our approach is a combination of concept terms analysis, using syntactic, lexical and contextual meanings of terms and tree mining algorithm presented in [22].

Our Contributions

1. Building clusters of similar terms based upon schema elements' labels similarity. The similarity is computed using label's syntactic, lexical and contextual (hierarchical) occurrence in the schema.
2. Mini-taxonomies extraction using tree mining for ontology concepts learning.
3. Verification of the semantic precision of the generated min-taxonomies.

¹ Web Service Modeling Ontology - <http://www.w3.org/Submission/WSMO/>

4. Similar hierarchical patterns generation from similar terms clusters.
5. Automatic production of trustable basic domain taxonomy from a given set of domain specific schema trees, implying domain community consensus over it.

The remainder of the paper is organized as follows. Section 2 presents the background of ontology concept and taxonomy. Section 3 gives the related work in the field of ontology learning. In Section 4 we describe our approach, describing the architecture and related algorithms. Section 5 demonstrates a running example to support our approach. Section 6 presents the experimental evaluation along with discussion on the results. Section 7 outlines future perspective and concludes.

2 Ontology Learning Background

Discussion on ontology building and utilisation has been around since early 90s. Ontology has been defined in [11] as an explicit, formal specification of a shared conceptualization of a domain of interest. Formalization aspect highlights the machine readability of the ontology and shared conceptualization points towards its acceptance by the players of the domain. Initial ontology development endeavors resulted in the form of DAML ² and OIL ³ languages. Today the features of the two languages have been extended to OWL ⁴ using XML based RDF schema ⁵.

Initial focus in ontology design has been the manual technique but with the passage of time more and more semi-automatic techniques have emerged, facilitated by ontology editing tools⁶. This semi-automatic approach is named as the ontology learning process.

Ontology learning is a combination of tasks organised as a layered approach, in the manner of increasing complexity. The tasks are enumerated by Paul Buitelaar et al. in [4] as term extraction, synonym and translation detection, concept formulation, concept hierarchies, relations, rule derivation and axiomatization.

Concept hierarchy, also called taxonomy (is-a relation), is a tree structure of classifications for a given set of ontological objects. It is considered to be the ontology backbone. At the top of this structure is a single classification, the root node, that applies to all objects. Nodes below this root are more specific classifications that apply to subsets of the total set of classified objects. So for instance, in common schemes of books, the root is called "Book" followed by nodes for the type: Art, Science, Fiction, Sports, etc. And each instance of "Book" concept can have properties like author, title, publisher etc. (Figure 1)

² DAML: Darpa agent Markup Language - <http://www.daml.org/>

³ OIL: Ontology Interface Layer - <http://www.ontoknowledge.org/oil/>

⁴ OWL: Web Ontology Language - <http://www.w3.org/TR/owl-features/>

⁵ RDF: Resource Description Framework - <http://www.w3.org/RDF/>

⁶ Protege is a free, open source ontology editor and knowledge-base framework; <http://protege.stanford.edu/>

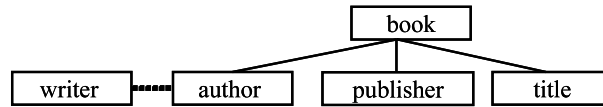


Fig. 1. Ontology taxonomy example.

In this paper we present work which is a step towards automatic conceptualisation of domain ontology for a certain domain which is already populated with user defined schema structures for diverse applications.

3 Related Work

One of the foremost techniques applied for ontology learning have been *term extraction* from text. Similar terms are clustered together for further analysis and inception of inter term relations or taxonomy. These methods have their roots in natural language processing research [5]. Buitelaar et al. present their OntoLT approach as a plug-in for protege ontology editing tool. The authors define preconditions using XPATH expressions over the XML based linguistic annotations. The rules help in constructing or extending an ontology. The preconditions revolve around the linguistic constructs in a sentence. For example if the subject in the sentence corresponds to a certain morphological stem of a word.

Terms similarity computation has been researched in two ways. Primarily by using readily available lexical resources like Wordnet⁷. And secondly, by devising clustering algorithms based on the similarity of terms' syntactic contexts. Term indexing based information retrieval techniques [19] and data mining methods [12] provide the space for such algorithms.

There is no definite definition available for *concept formation*. Our approach follows the hierarchical representation of concepts [8] which can be extended, upon receiving further information about the concept. The extension idea has been pruned in [15] as binary relation extraction of terms and recommendations have been made for use of data mining co-occurrence algorithms. These methods can ultimately provide an incremental approach for ontology learning.

Since semantic Web is the biggest gainer in the research of ontologies, web has also been extensively exploited in this regard. [21] describe a tool which prunes the web resources like Wikipedia, Wiktionary, along with domain corpus for domain ontology learning. These resources are exploited against a set of candidates extracted from a set of ontology instances using the linguistic context. Another work by Maedche et al. [14] explains two algorithms top-down and bottom-up approaches, for deducing taxonomic relations from the web based on heuristics. Our approach presents a similar top-down method, by applying tree-mining on the available hierarchical structures in a domain. In [10], the authors

⁷ <http://wordnet.princeton.edu>

present the use of semi-structured schemata (XML and RDF based resources) for constructing a domain ontology, manually and semi-automatically.

Another interesting research for ontology generation is the use of tables extracted from web and other resources. Authors in [20] argue that extracting relational knowledge from tables is much easier than exploiting the text corpus. The research describe a comprehensive framework for assembling human created tables. The approach canonicalises the table information, generates a mini-ontology from it and then incrementally merges the mini-ontologies.

4 Our Approach for Mining Ontology Concepts and Taxonomy

In this section we present our approach for detection of ontological concepts, as a hierarchical structure, from available domain specific schema tree structures. We discuss the architecture and the related algorithms in length to clarify the novelty of our method.

4.1 Definitions

Here we give the basic definitions supporting the implementation for our technique.

Definition 1 (Hierarchical Structure): A Hierarchical Structure $S = (V, E)$ is a rooted, labelled tree [22], consisting of nodes $V = \{0, 1, \dots, n\}$, and edges $E = \{(x, y) \mid x, y \in V\}$. One distinguished node $r \in V$ is called the root, and for all $x \in V$, there is a unique path from r to x . Further, $lab: V \rightarrow L$ is a labeling function mapping nodes to labels inling function mapping nodes to labels in $L = \{l_1, l_2, \dots\}$.

In further text we will refer to hierarchical structure as tree. Tree nodes bear two kinds of information: the node label, and the node number allocated during depth-first traversal. Labels are linguistically compared to calculate label similarity (Definition 2, Label Semantics). Node number is used to calculate the node's tree context (Definition 3, Node Scope).

Definition 2 (Label Semantics): A label l is a composition of m strings, called tokens. We apply the tokenisation function tok which maps a label to a set of tokens $T_l = \{t_1, t_2, \dots, t_m\}$. Tokenisation [9] helps in establishing similarity between two labels.

$tok : L \rightarrow \mathcal{P}(T)$, where $\mathcal{P}(T)$ is a power set over T .

Example 1 (Label Equivalence): 'FirstName', tokenised as {first,name}, and 'NameFirst', tokenised as {name, first}, are equivalent, with 100 % similarity.●

Label semantics corresponds to the meaning of the label (irrespective of the node it is related to). It is a composition of meanings attached to the tokens

making up the label. As shown by Examples 1 and 2, different labels can represent similar concepts. We denote the concept related to a label l as $C(l)$.

Example 2 (Synonymous Labels): ‘WriterName’, tokenised as {writer,name}, and ‘AuthorName’, tokenised as {author, name} are equivalent (they represent the same concept), since ‘writer’ is a synonym of ‘author’.

Definition 3 (Node Scope): In tree S each node $x \in V$ is numbered according to its order in the depth-first traversal of S (the root is numbered 0). Let $T(x)$ denote the sub-tree rooted at x , and x be numbered X , and let y be the rightmost leaf (or highest numbered descendant) under x , numbered Y . Then the scope of x is $scope(x)=[X, Y]$. Intuitively, $scope(x)$ is the range of nodes under x , and includes x itself, see Figure 2. The count of nodes in $T(x)$ is $Y - X + 1$.

4.2 Scope Properties

Scope properties describe the contextual placement of a node [22]. Property testing involves simple integer comparisons. We utilise these properties in frequent sub-tree detection (details in sections 4 and 5).

Given $x [X,Y]$, $xd[Xd,Yd]$, $xa[Xa,Ya]$, and $xc[Xc,Yc]$:

Property. 1: Descendant (x,xd), xd is a descendant of x : $Xd > X \wedge Yd \leq Y$

Property. 2: Ancestor (x,xa), complements Property. 2, xa is ancestor of x : $Xa < X \wedge Ya \geq Y$

Property. 3: Cousin (x,xc) with non-overlapping scope, xc is cousin of x : $Xc > Y$.

Example 3 (Scope Properties Use) : Let us consider Fig. 2. We perform the descendant node check on nodes [2,2] and [5,5] with respect to **writer**[1,2]. Node [2,2] is a descendant of [1,2], using Property 1, and node [5,5] is not a descendant of [1,2]. Conversely speaking **writer**[1,2] is an ancestor of node [2,2] and not of node [5,5] according to property 2. Consider node **writer**[1,2] and node **publisher**[4,5]. The two nodes are cousin nodes since they satisfy the Property 3.

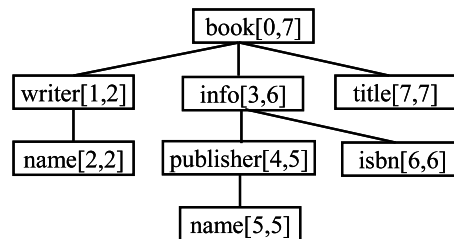


Fig. 2. Input hierarchical structure with scope.

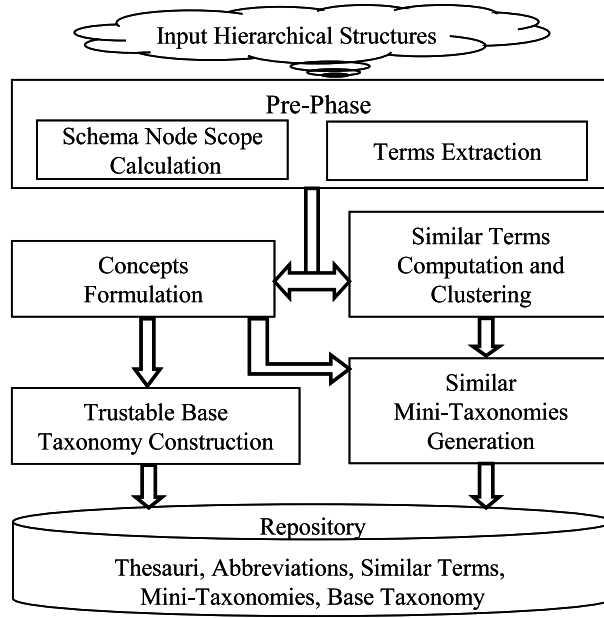


Fig. 3. Architecture for tree mining ontology concepts and taxonomy.

4.3 Architecture

The architecture of our approach for ontology taxonomy learning through tree mining is shown in Figure 3. The approach is composed of five modules: (i) *Pre-Phase*, (ii) *Similar Terms Computation and Clustering*, (iii) *Concepts Formulation*, (iv) *Similar Mini-Taxonomies Generation* and (v) *Trustable Base Taxonomy Construction*, supported by a repository which houses oracles and concepts' taxonomies.

The system is fed a set of hierarchical structures (XML Schema instances). *Pre-Phase* module processes the input as trees, calculating the depth-first node number and scope (Definition 3) for each of the nodes in the input schema trees. At the same time, for each schema tree a listing of nodes is constructed, sorted in depth-first traversal order. As the trees are being processed, a sorted global list of distinct terms (node labels) over the whole set of input is created (details in [18]).

In *Similar Terms Computation and Clustering* module, similarity is derived after analysing the tree node labels. We tokenise the labels and expand the abbreviated tokens using an abbreviation oracle. Currently, we utilise a domain specific user defined abbreviation table. Further, we make use of token similarity, supported by an abbreviation table and a manually defined domain specific synonym table. Label comparison is based on similar token sets or similar synonym token sets. The architecture is flexible enough to employ additional abbreviation or synonym oracles or arbitrary string matching algorithms. To further

refine the similarity, we employ the structural aspect also. Labels instances at nodes in different trees are compared for ancestor level label instance similarity (Property 2). Any such existence helps in re-enforcing the similarity of current pair of labels and remove any ambiguity [18]. Based on the similarity, the terms are clustered together.

In our approach concept is considered to be small tree structure which we call a Mini-Taxonomy. In *Concepts Formulation* module, such concepts are extracted. We utilise frequent sub-tree mining approach described in [22] for this purpose. Our algorithm ExSTax⁸, an extended version of this approach, acts as the kernel of our system. The algorithm is explained in next sub-section.

Once the set of mini-taxonomies have been extracted, this set is fed to the *Similar Mini-Taxonomies Generation* module. At this stage all possible similar mini-taxonomies are generated with the help of already computed similar terms clusters. In parallel, from the *Trusted Base Taxonomy Construction* module, a taxonomy is generated. In fact it is the final iteration of the ExSTax algorithm, which results in a set of largest possible frequent sub-trees. If there is just one tree, we consider it as the base taxonomy else all the sub-trees are merged together to produce the base taxonomy.

The *Repository* is an indispensable part of the system. It houses oracles: thesauri and abbreviation lists. It also stores extracted terms, inter-term similarity, mini-taxonomies representing concepts and trustable base taxonomy. And it provides persistent support to the taxonomy learning process.

4.4 ExSTax Algorithm

The ExSTax algorithm presents an iterative nature based on incrementally extracting frequent sub-trees from a given set of trees. The sub-tree frequency in the forest of trees is user defined parameter. The algorithm takes as input the list of terms, with similar terms linked together to form a cluster (each cluster can have one or more terms). First task performed by the algorithm is to compute the frequency of each term in the forest of trees. Next, within each cluster, the term with the highest frequency in the forest of trees is taken as the symbol representing the cluster. The frequency of the cluster symbol is computed by adding frequencies of all the terms in the cluster. From here on the algorithm executes similar to frequent sub-tree mining algorithm given in [22], with cluster symbols as the starting labels for the vertical-join-list data structure (explained in section 5).

Firstly, the process finds frequent sub-trees with size 1, and creates the list data structure for further joining. Only subtrees with frequency equivalent or greater than the threshold are kept in the list. In second pass, a new list of join lists is created. Each frequent size 1 sub-tree is joined with every other size 1 sub-tree in the first list. If the pair passes the Property 2 i.e. descendant test is true for the pair, a new label for the sub-tree size 2 is created. The new sub-tree

⁸ ExSTax is an anagram of the bold letters in the phrase: Automatic **E**xtraction of Structurally Coherent Mini-**T**axonomies

label is added to the second list, along with its join-list entry . Subsequent size 2 sub-trees are added to the list. At end of the iteration for size 2, frequency of each sub-tree is computed and only sub-trees with equivalent or higher frequency then threshold are kept in the list. Threshold frequency is computed as " *support multiplied by number of input schemas divided by hundred* ". The iterative process keeps executing till the sub-tree list does not have any frequent sub-tree. For joining sub-tree size 2 or greater, property 3 (cousin test) is also evaluated for computing a sub-tree label.

The last list of sub-trees contain either one or more sub-trees. This list acts as the input for computing the base taxonomy for the given set of hierarchical structures. The sub-trees lists, along with similar terms clusters are input to the module for Similar Mini-Taxonomies Generation.

5 Running Example

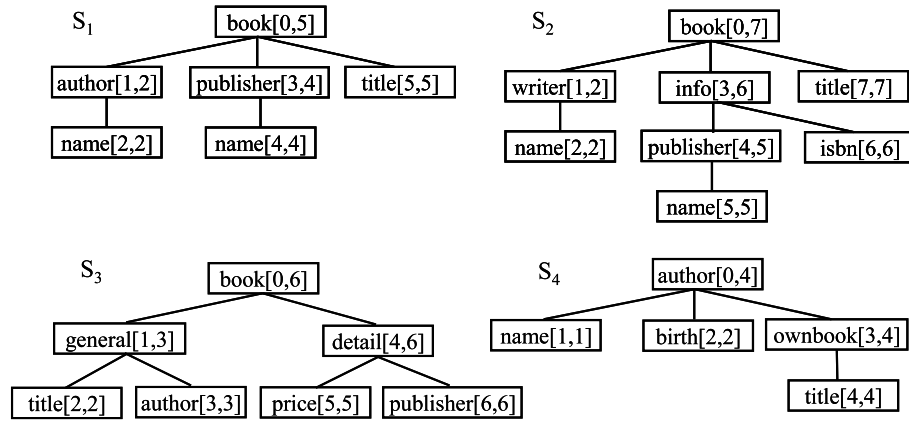


Fig. 4. Input set of 4 trees for learning base taxonomy using tree mining.

Figure 4 shows four trees after Pre-Phase. A list of terms created in this traversal is enumerated in Figure 5a with the similar terms clusters. Incremental execution of ExSTax algorithm is demonstrated in Figure 5b . There are six iterations before the algorithm stops, when it is not possible to generate much larger frequent sub-tree. The sub-tree generated in the last iteration can be considered as the base taxonomy for the given set of hierarchical structures. Figure 6 illustrates the taxonomical structure generated.

The six iterations are presented in the six panels of Figure 5b. First iteration takes into account sub-trees of size one. Since there is no right prefix tree, the prefix data structure is empty. Each sub-tree label's vertical list element is paired with other labels' vertical list elements. The joining of vertical lists result in a

a. Terms List with frequency and Similar clusters (author, writer), (book, ownbook) and (detail, info)												
author	book	birth	detail	general	info	isbn	name	ownbook	price	publisher	title	writer
2	3	1	1	1	1	1	5	1	1	3	4	2
b. Labels of sub-tree of size 1-6 with frequency greater then threshold and vertical join lists {tree number, prefix sub-tree, [number, scope (of right most node)]}												
author	book	detail	name	publisher	title							
1,,[1,2]	1,,[0,5]	2,,[3,6]	1,,[2,2]	1,,[3,4]	1,,[5,5]							
2,,[1,2]	2,,[0,7]	3,,[4,6]	1,,[4,4]	2,,[4,5]	2,,[7,7]							
3,,[3,3]	3,,[0,6]		2,,[2,2]	3,,[6,6]	3,,[2,2]							
4,,[0,4]	4,,[3,4]		2,,[5,5]		4,,[4,4]							
			4,,[1,1]									
* - indicates depth-wise downward move and / upward move in the tree structure												
author-name	book-author	book-detail	book-name	book-pub	book-title	detail-pub	pub-name					
1,1,[2,2]	1,0,[1,2]	2,0,[3,6]	1,0,[2,2]	1,0,[3,4]	1,0,[5,5]	2,3,[4,5]	1,3,[4,4]					
2,1,[2,2]	2,1,[1,2]	3,0,[4,6]	1,0,[4,4]	2,0,[4,5]	2,0,[7,7]	3,4,[6,6]	2,4,[5,5]					
4,0,[1,1]	3,0,[3,3]		2,0,[2,2]	3,0,[6,6]	3,0,[2,2]							
			2,0,[5,5]		4,3,[4,4]							
book-author/detail, book-author-name, book-author/name, book-author/pub, book-author/title, book-detail/pub, book-name/pub, book/name/title, book-pub-name, book-pub/title												
book-author/detail-pub, book-author-name//name, book-author-name//pub, book-author-name//title, book-author/name/title, book-author/pub-name, book-author/pub/title, book-name/pub/title, book-pub-name//title												
book-author-name//name/title, book-author-name//pub-name, book-author-name//pub/title, book-author/pub-name//title												
book-author-name//pub-name//title 1,01234,[5,5] 2,01245,[7,7]												

Fig. 5. List of terms with join lists for frequent sub-trees of size 1-6 with 50% frequency.

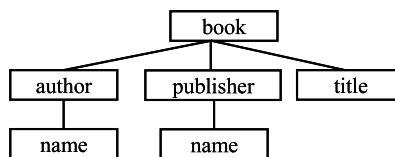


Fig. 6. Trusted extracted taxonomy.

structure of size two i.e. one subtree can only be descendant of the other in this case. The sub-trees of size two and which are present in atleast two of the input structures (50% support), are added to the second list. In vertical list element, last prefix entry denotes the number of the right most node of the prefix sub-tree (Figure 5b).

In subsequent iterations, both descendant test (Property 1) and cousin test (Property 3) are applied to come up with frequent sub-trees. Panels 3-5 present the labels of extracted frequent sub-trees (mini-taxonomies). The last panel of sub-tree gives sub-tree with six elements. There are two vertical list elements, supporting the 50% support condition. The sub-tree is present in input structures 1 and 2 (Figure 4). With in each vertical list element, the prefix structure gives the list of node numbers of nodes present in the sub-tree prefix.

6 Evaluation

The prototype implementation uses Java 5.0. A PC with Intel Xeon, 2.33 GHz processor and 2 GB RAM, running Windows XP was used. We selected a set of synthatic XML schema trees as the input heirarchical structures for our experiments. The data set has 176 schema tree instances, with size in nodes: maximum 14, minimum 5, average 8 and maximum depth in any schema equal to 3.

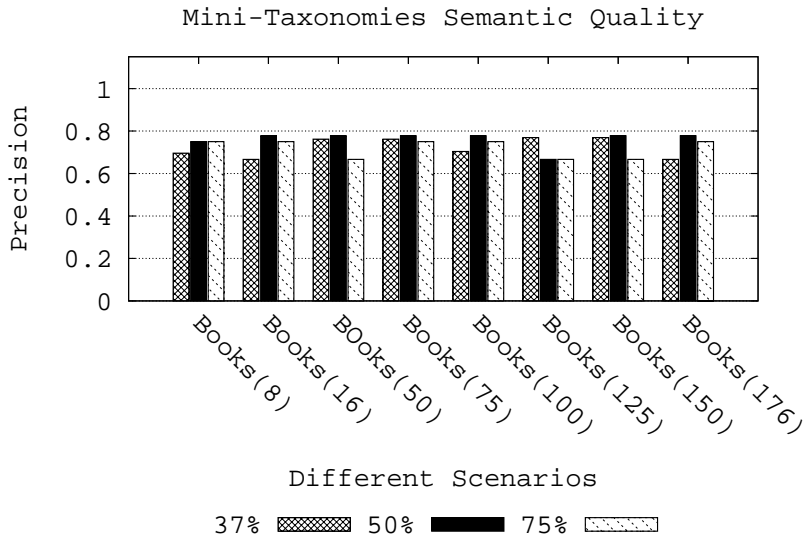


Fig. 7. Precision of ExSTax for eight sets of hierarchical structures.

We examine the semantic quality of generated mini-taxonomies using the precision measure. Our target is to generate semantically meaningful taxonomic structures. Therefore, we manually scrutinized the generated tree patterns and computed the share of semantically applicable sub-trees among all

found. With reference to Figure 4 structure S_1 , a sub-tree structure "book[0,5]-name[2,2]/name[4,4]" is considered to be invalid, since it is semantically meaningless. Based on these considerations we show the precision measure computed from the experiments. Figure 7 shows the precision of 8 sets of input structures comprising of 8, 16, 50, 75, 100, 125, 150 and 176 sizes taken from BOOKS. The results are computed for three different tree mining support values 37, 50 75 percent.

Discussion

The experimental results show the precision measure to be between 0.65 and 0.8. Thus supporting the validity of our idea of mini-taxonomies extraction. The number of mini-taxonomies generated increased with decrease in the value of tree mining support parameter and vice versa. Therefore we selected the support values range (37-75), whose results could be varified manually. Secondly, it is quite difficult to estimate the recall measure in the experiments because of the large number of possible outputs. Devising a system for this purpose is out of the scope of current work. Another observation made during the execution is that ExSTax algorithm shows exponential scalability with respect to the size of input tree structures. Since we are concerned with the semantic validity of the output, we did not took into account the time performance complexity of the algorithm.

7 Conclusion and Future Work

We have introduced a novel technique based on tree mining, for ontology taxonomy learning. The core idea behind this paper is to demonstrate the applicability of tree mining techniques for ontology taxonomy extraction in large scale scenario. The technique inherently supports the collaborative ontology learning by holistically exploiting the already available hierarchical structures in the domain.

We have investigated its scalability with respect to number of schemas. The experimental results demonstrate that our approach scales to hundreds of schemas. The linguistic matching of node labels uses tokenisation, abbreviations and synonyms. Our method provides an almost automated solution to the large scale taxonomy learning problem.

Our results point to significant future research. Foremost work tends to be the research for knowledge of valid patterns missing from the generated set, to estimate the recall measure. We are planning to investigate the application of our approach in P2P architectures, and enhancements to heuristics based term matching. Another issue for the future is a benchmark for automatic ontology learning tools in a large scale scenario. To further benefit from tree mining, we are going to use the automatically extracted mini-taxonomies for the discovery of n:m complex mappings in context of research described in [8].

References

1. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, 2004.
2. I. B. Arpinar, B. Aleman-Meza, R. Zhang, and A. Maduko. Ontology-driven web services composition platform . In *IEEE CEC*, 2004.
3. P. Buitelaar. Ontology learning: Where are we? and where are we going? - invited talk. In *ISWC WorkShops ESOE*, 2007.
4. P. Buitelaar, P. Cimiano, and B. Magnini. Ontology learning from text: An overview. In *Ontology Learning from Text: Methods, Evaluation and Applications Frontiers*. IOS Press, 2005.
5. P. Buitelaar, D. Olejnik, and M. Sintek. A protege plug-in for ontology extraction from text based on linguistic analysis. In *ESWS*, 2004.
6. Y. Chi, R. R. Muntz, S. Nijssen, and J. N. Kok. Frequent subtree mining - an overview. *Fundamenta Informaticae*, 66(1-2):161–198, 2005.
7. P. Cimiano, A. Pivk, L. Schmidt-Thieme, and S. Staab. Learning taxonomic relations from heterogeneous sources of evidence. In *ECAI Workshop Ontology Learning and Population*, 2004.
8. D. W. Embley, L. Xu, and Y. Ding. Automatic direct and indirect schema mapping: Experiences and lessons learned. *ACM SIGMOD Record*, 33(4):14–19, 2004.
9. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic matching. In *ESWS*, 2004.
10. A. Gomez-Perez and D. Manzano-Macho. Deliverable 1.5: A survey of ontology learning methods and techniques. Technical report, Universidad Politecnica de Madrid, 2003.
11. T. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *Human and computer Studies J.*, 43:907–928, 1994.
12. B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *KDD*, pages 148–157, 2004.
13. M. Li, X.-Y. Du, and S. Wang. Learning ontology from relational database. In *IEEE ICMLC*, 2005.
14. A. Maedche, V. Pekar, and S. Staab. Ontology learning part one – on discovering taxonomic relations from the web. In *Web Intelligence*, 2002.
15. A. Maedche and S. Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook of Ontologies*. Springer Verlag, 2004.
16. N. F. Noy. Semantic integration: A survey of ontology-based approaches. *ACM SIGMOD Record*, 33(4):65–70, 2004.
17. C. Parent and S. Spaccapietra. Database integration: The key to data interoperability. In M. P. Papazoglou, S. Spaccapietra, and Z. Tari, editors, *Advances in Object Oriented Modeling*. The MIT Press, 2000.
18. K. Saleem, Z. Bellahsene, and E. Hunt. Porsche: Performance oriented schema mediation. *Information Systems - Elsevier*, 33, 2008.
19. H. Schutze. Word space. In *NIPS*, pages 895–902, 1993.
20. Y. A. Tijerino, D. W. Embley, Y. Ding, and G. Nagy. Towards ontology generation from tables. *World Wide Web*, 8:261–285, 2005.
21. N. Weber and P. Buitelaar. Web-based ontology learning with isolde. In *ISWC WorkShops Web Content Mining with Human Language*, 2006.
22. M. J. Zaki. Efficiently mining frequent embedded unordered trees. *Fundamenta Informaticae*, 66(1-2):33–52, 2005.