A NAIVE METHOD FOR ONTOLOGY CONSTRUCTION

Preeti Kathiria[1] and Sukraat Ahluwalia[2]

[1]Assistant Professor, Department of Computer Science and Engineering, Nirma University, Gujarat, India

[2]Student, Department of Computer Science and Engineering, Nirma University, Gujarat, India

ABSTRACT

Ontologies are being used to organize information in many domains like artificial intelligence, information science, semantic web, library science. Ontologies of an entity having different information can be merged to create more knowledge of that particular entity. Ontologies today are powering more accurate search and retrieval in websites like Wikipedia etc. As we move towards the future to Web 3.0, also termed as the semantic web, ontologies will play a more important role.

Ontologies are represented in various forms like RDF, RDFS, XML, OWL etc. Querying ontologies can yield basic information about an entity. This paper proposes an automated method for ontology creation, using concepts from NLP (Natural Language Processing), Information Retrieval and Machine Learning. Concepts drawn from these domains help in designing more accurate ontologies represented using the XML format. This paper uses document classification using classification algorithms for assigning labels to documents, document similarity to cluster similar documents to the input document, together, and summarization to shorten the text and keep important terms essential in making the ontology. The module is constructed using the Python programming language and NLTK (Natural Language Toolkit). The ontologies created in XML will convey to a lay person the definition of the important term's and their lexical relationships.

KEYWORDS

Ontology, Semantic Web, Domain Ontology, Ontology Construction, Classification, Document Similarity.

1. INTRODUCTION

Ontology is primarily a philosophical concept, the philosophical definition of ontology is “The study of the nature of existence of an entity and it's relationships”. Computer Science extends this definition of ontology by adding a taxonomy. The taxonomy can is based on the entity. A good example of the ontology of an entity is given below from a discussion on semantic web.

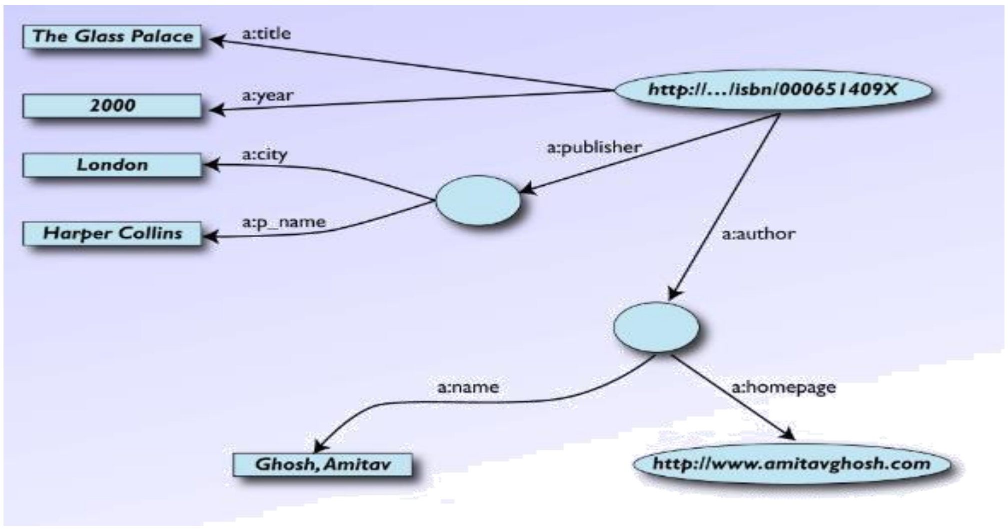
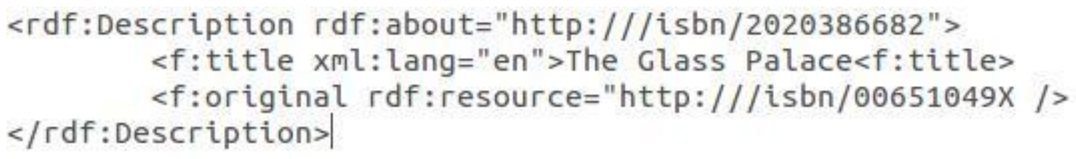


Figure 1: Ontological representation of the English version of the book.[1]

The above picture is an ontology constructed for the book “The Glass Palace” written by Amitav Ghosh. The ontology is visualized as a graph where each of the parts of an entity, in this case the book, such as the city (London), Authors' name (Amitav Ghosh), all form the nodes of the graph. An RDF triple representation of the ontology as illustrated in [1] would be:



```
<rdf:Description rdf:about="http:///isbn/2020386682">
        <f:title xml:lang="en">The Glass Palace<f:title>
        <f:original rdf:resource="http:///isbn/00651049X />
</rdf:Description>
```

Figure 2: RDF representation of the ontology.[1]

Querying the information above can give vital information like the author of the book, the publisher etc. With ontology as a method for organizing information, extraction of basic information of an entity becomes much more easier. Now consider another ontology of the book but with a different author for the translated version of the book. The graph as shown in [1] would be:

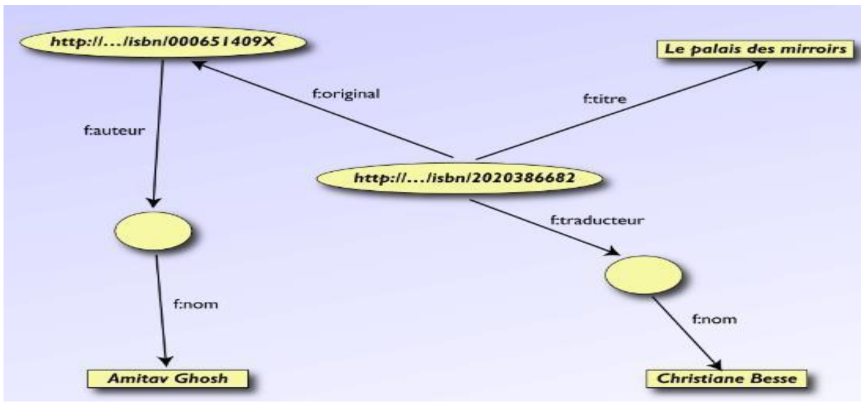


Figure 3: Ontological representation of French version of the book.[1]

Since the two ontologies above have the same URI, they can be merged to form a singular ontology of the entity, containing more information now. Querying the new ontology now yields more information than before. A merged ontology would be diagrammatically represented as illustrated in:

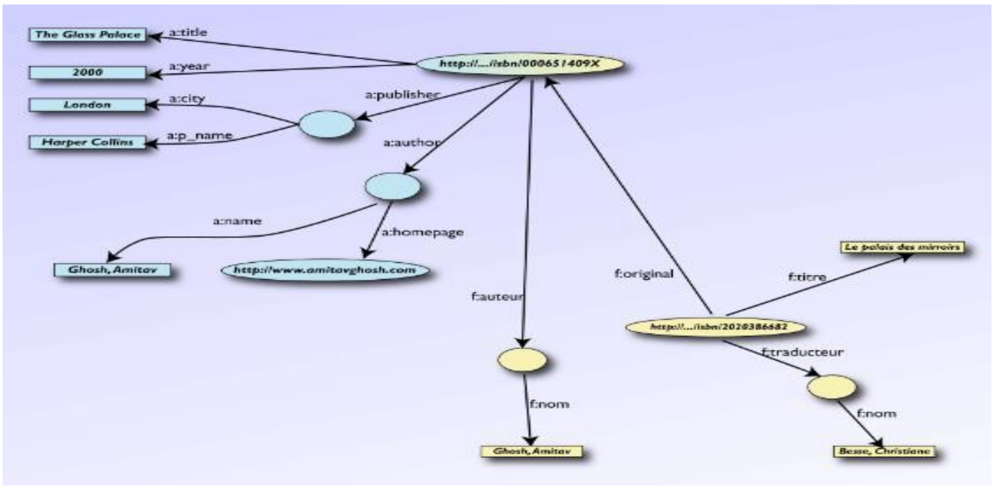


Figure 4: Merged Ontology.[1]

The advantages of using ontologies can be summarized from [2]:

Ontologies provide a specific vocabulary with which knowledge can be represented. This vocabulary allows us to specify which entities will be represented, how they will be represented, how they can be grouped, and what relationships connect them together. The vocabulary can be seen as a sort of social contract between a data producer and a data consumer. The more accurate the ontology, the greater the potential understanding of how data can be used, If an ontology is complex, including relationships a data consumer doesn't require then it can become confusing, complicated, and difficult to use and maintain.

Ontologies allow us to express the formal rules for inference. When software reads our ontology, it should have all the information necessary to draw the same conclusions from our data that we did. When actions of software are consistent with the rules expressed in ontology, we say that the software has made an ontological commitment. In object-oriented programming (OOP), it is easy to forget how many rules the system is handling for us. For example, a subclass in an OO system contains its parent's members (methods and variables) because, somewhere in the system, software asserted that having access to parent members is consistent with what it means to be an OO subclass. By design, semantic systems using ontology, leave it up to you to build or invoke the necessary software to make inferences based on that model.

2. RELATED WORK

Researchers have tried to construct ontologies through different means:

1) Manually using tools like Protege, OWL etc.
2) Using automated methods drawing techniques from areas like NLP, Machine Learning, and Information Retrieval etc.

An example that follows the second technique is employed by Q.Song, J. Liu, X. Wang and J.Wang [3] , employs techniques from data mining and information retrieval. It uses K Means clustering to construct the feature vector using tf-idf of each page. They further use Bayesian classifiers on sample data sets to construct domain ontologies. S.Hong Sie and J.Hua Yeh[4], have proposed an automated construction method based on a hierarchy generation process with a guided concept clustering method(GCC), to reduce human effort in maintaining large knowledge networks.

Among the methods that propose creation of ontology using manual methods, the method proposed by L.Seremati and A. Kameas[5], is targeted towards novice ontology developers with little experience in developing ontologies. The ontology developer in their method, has experience in the domain for which ontology is being made, the ontology construction is done by re-using pre-existing ontologies. The second case in manual construction is domain based in the area of biomedical ontologies and has been proposed by F.K Hussain,A.S Sidhu, T.S Dhillon, E. Chang[6]. The method takes a case study of protein ontologies, constructing ontologies by taking protein data from existing data sources. The design of the ontologies is distributed across sections like SubClass,ATOMsequence of protein. The final method of manual construction proposed by S. Jain[7] takes into account the relation of a topic to other topic, for eg. a topic of computer science may also be related to domains of Mathematics, Statistics, Electrical Engineering etc. The method uses OWL to construct domain ontologies of various topics and sub-topics covered in a document.

3. ONTOLOGY CONSTRUCTION MODULE

3.1. Document Classification

The very first step in the construction phase is classifying the type of the document. To construct a domain ontology, it is important that a label type be assigned to the document, so that the correct relation can be assigned. To do classification in the construction phase Naive-Bayes classification shall be used.

3.1.1. Naive-Bayes Classification

Naive Bayes classification per its name works on bayesian probability. Bayes formula can be represented as:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

Figure 5: Bayes Formula.[8]

P(A) and P(B) are the the probabilities of A and B without regard to each other. P(A|B) is the probability of occurrence of A given that B is true. In Naive Bayes classification the principle applied is that every label has a say. This means that out of the set of possible labels for an entity, the probabilities of the entity belonging to all labels is calculated using the above stated formula. Therefore if an entity could belong to N possible labels represented by a vector *x = (x1,...,xn),* , then the formula for classification using Naive Bayes can be represented as:

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

Figure 6: Formula for Naïve Bayes classification.[9]

The above formula is suitable when the data is discrete. If the data is continuous for example the classifier might have to identify the gender based on real attributes like height, weight, foot size, attributes which can be in floating point format, the above formula will not work. In cases like that, values of each class are distributed according to a Gaussian distribution, given by the formula:

$$p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Figure 7: Naive-Bayes for continuous data.[8]

Naive Bayes is a lazy learner among classifiers. Naive Bayes requires to be first run against a set of training data where it calculates probabilities for different features that belong to particular labels. Once that is done, the data to be classified is fed in and the probabilities for its features are calculated for each label, using the results of the training set. The label which has the maximum probability for occurrence, is assigned to the data.

3.1.2. Using Naive-Bayes to classify documents

The aim of the Ontology construction module is to classify a document as belonging to either Mathematics or Computer Science. At first a training set of documents belonging to both domains are input to the classifier, the label of the document is also input to the classifier. The classifier uses a feature vector, *W = (W1,....,Wn),* to represent words of the document. The probability of the word occurring is mapped to the label of the training set documents. Once this process has been done, the document to which the label has been assigned is split into the same feature vector as the training set documents and its probabilities are calculated based on the results of the training set for both domains - Mathematics and Computer Science. The domain which has the higher probability, is the label that is assigned to the document. Once classification of a document is complete, the next step is to search the most similar documents.

3.2. Document Similarity

After classification, the next step is to search for documents that are similar to the document for which the ontology will be constructed. The method to be used is the vector space model.

3.2.1. Vector Space Model

The vector space model begins by creating the bag of words of two or more documents. Considering two documents A and B, the bag of words of both documents is a vector of the collection of unique words in both the documents. A vector *V = (V1,...,Vn),* is constructed for both documents, the vector constitutes of the count of occurrence of each word in the documents that corresponds to the bag of words. The vector can also be called a term frequency vector, abbreviated as *tf,* The next step is to calculate the inverse document frequency or *idf* , of the documents. The inverse document frequency of a document is a measure of how important the word is to the document, whether it is common or rare across a set of documents. The idf of a document is useful in filtering out stop words, words that occur very frequently but do not add much meaning to the document. The inverse document frequency can be calculated by:

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Figure 8: Inverse Document Frequency. [10]

Where N is the number of documents in the dataset and D is the document in which the term t belongs to. Once this value has been calculated the similarity is given by:

$$\cos\theta = \frac{\mathbf{d_2} \cdot \mathbf{q}}{\|\mathbf{d_2}\| \, \|\mathbf{q}\|}$$

Figure 9: Cosine similarity for computing document similarity.[10]

The cosine similarity of the document is calculated with respect to all documents in the corpus and the top five documents are selected. The text of all the five documents is merged along with the original document and is followed by summarization.

3.3. Summarizing the text

The next step is that of summarization. Summarization has the benefits of reducing the text size to be analyse and also returns the important keyword's of the document. The algorithm as proposed in [11] is as described below:

1) Segment the corpus into individual sentences.
2) Compare the intersection of a sentence with all other sentences. An intersection here means the count of the common words in the document divided by 2(total sentences being compared). Assign an intersection score based on the formula:

$$\frac{Score = Words(S1 \cap S2)}{2}$$

Figure 10: Formula for computing intersection score.

3) The above result is stored for each sentence in a 2-D array. What one ends up with is a graph showing the commonness of one sentence with other sentences in the text.
4) The intersection score for all sentences is added up.
5) Add the top sentence with highest score from each paragraph to the summarized text. To increase the accuracy cosine similarity is used:

The above algorithm when used contracts the size of the text by almost 60% on an average without much loss of information. The first task post summarization is to firstly remove all stop words from sentences in the summarized text. Stop words constitute of words like a, an, and the etc. Stop words are important in the context of deciphering the meaning of the sentence in text mining[6]. In the problem stated here stop words are not very helpful. In a text corpus words like and, the, have a very high frequency of occurrence and should not be considered while taking term frequency of documents, as will be discussed in the paragraph below. Post summarization the top N(where N = selected number of words for ontology creation) words are ranked using a TF(Term Frequency) like index. The words are ranked based on the number of times they appear in the text compared to other words. The ontologies of these N words (in this case five, for testing) is constructed.

Ontology construction after summarization is based on lexical relationships of important keywords. To derive the lexical relationships of an entity WordNet is used. WordNet developed by the Cognitive Sciences Laboratory at Princeton, consists of words ordered by their lexical relationships, into groups of synonyms called as synsets. WordNet classifies the words into is-a relationships. WordNet is sometimes attributed as a lexical ontology due to the hypernym/hyponym relationships present among synsets.

The following lexical relationships are taken into consideration when constructing our word ontology:

1) Synonyms: A synonym is a word or phrase that has exactly the same meaning of another word in the same or another language, For example the words defer and procrastinate, mean the same thing in the English language. Synonyms are an important lexical relation to be used since it is important to know the words that are similar to a word being analyzed and share the same meaning with that word.

2) Hyponym: Hyponym is a word of more specific meaning than a general or superordinate term applicable to it. For example, spoon is a hyponym of cutlery. Therefore if the idea of concept hierarchies is used then hyponyms of a word can be labelled as going down the concept hierarchy. Hyponyms are useful in the sense that they can give immediate examples of a word.

For eg: The hyponyms of the word “motor car” give the result set as various types of auto-mobile like a convertible, electric car and so on, therefore using these hyponyms we are “drilling down” to the subclasses of a word.

3) Hypernym: Hypernym is a word or phrase whose referents form a set including as a subset the referents of a subordinate term. For example the hypernyms of motor car would be entity, physical object, wheeled vehicle. Therefore hypernymy can be categorized as “rolling up” the concept hierarchy and going to the superclass of the object or the origin of the word. Using the relationships of hypernymy and hyponym we can see that the basic criteria of object and its relationships is being maintained while giving detailed knowledge of the word at the same time.

The relationships of the word are then ordered in XML format giving the information regarding relationships, definition of an entity as described in an ontology. Using these ontologies in dictionary based applications will enhance the information obtained as not only the meaning of the word is returned but also words similar to it, words that it can originate from and words that originate from it are retrieved.

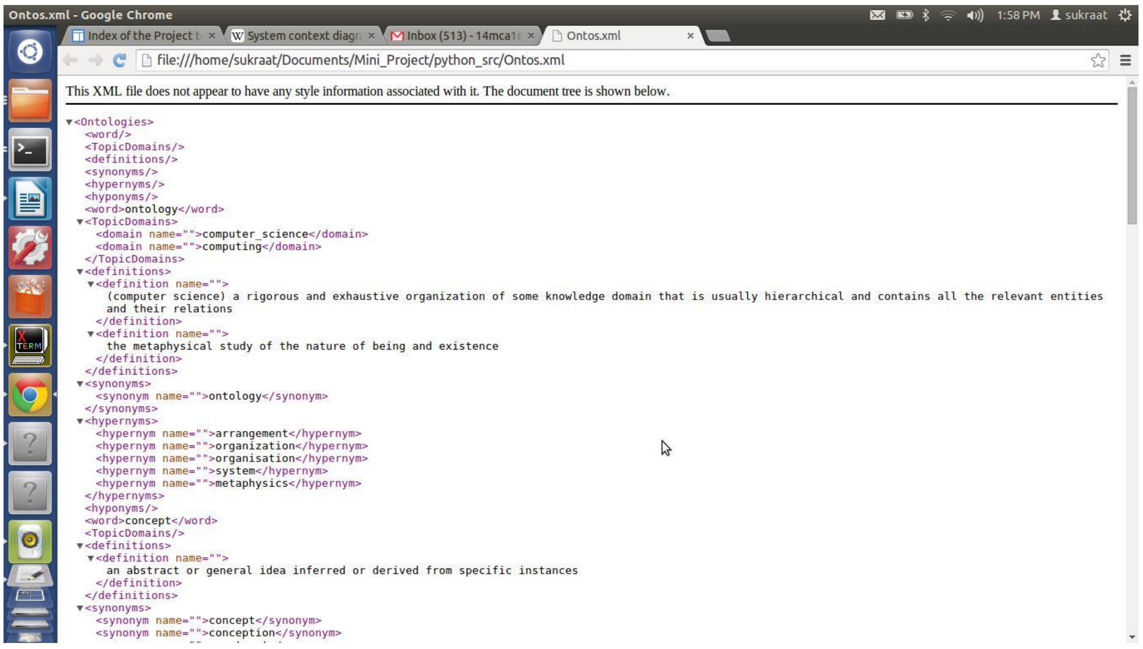
4. TESTING

To test the module documents from different domains were taken. Documents were tested based on the number of different lexical relations generated for the top five words in each document. The results have been described in the table below, in the table below, the numbers corresponding to sections - Definitions, Synonyms, Hypernyms and Hyponyms, refer to the number of lexical relation found for each word and added as part of the ontology:

Word	Definitions	Synonyms	Hypernyms	Hyponyms
Ontology	2	1	5	0
Document	6	4	15	48
Keyword	0	0	0	0
Concept	1	3	2	28
Relations	7	19	23	50

Table 1: Results of Ontology components after running ontology module.

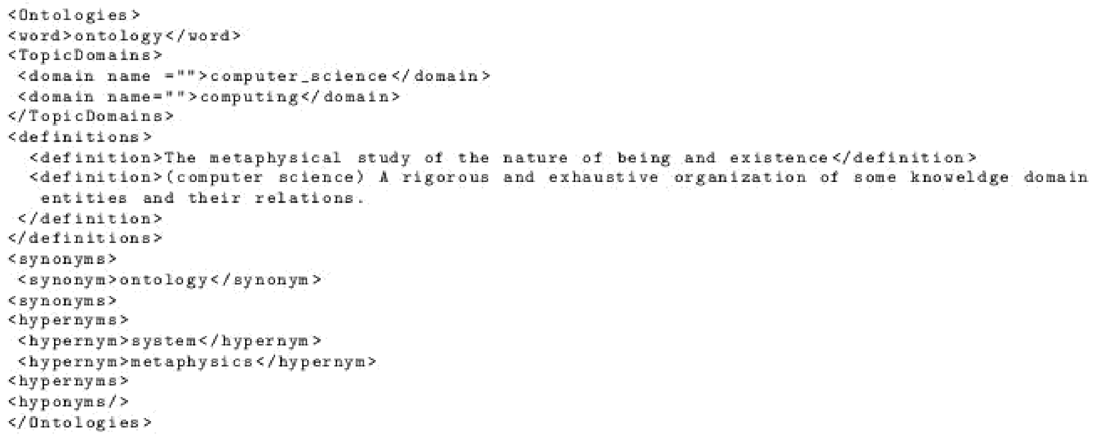
The above results in the table signify the total number of lexical relations that could be derived from WordNet for the top words after summarizing the text. The total number describes the size of each component of the ontology that has been formed.


Ontos.xml - Google Chrome
1:58 PM sukraat
Index of the Project b
System context diagr
Inbox (513) - 14mca1
Ontos.xml
file:///home/sukraat/Documents/Mini_Project/python_src/Ontos.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Ontologies>
   <word/>
   <TopicDomains/>
   <definitions/>
   <synonyms/>
   <hypernyms/>
   <hyponyms/>
   <word>ontology</word>
 ▼<TopicDomains>
    <domain name="">computer_science</domain>
    <domain name="">computing</domain>
  </TopicDomains>
 ▼<definitions>
  ▼<definition name="">
     (computer science) a rigorous and exhaustive organization of some knowledge domain that is usually hierarchical and contains all the relevant entities
     and their relations
    </definition>
  ▼<definition name="">
     the metaphysical study of the nature of being and existence
    </definition>
  </definitions>
 ▼<synonyms>
    <synonym name="">ontology</synonym>
  </synonyms>
 ▼<hypernyms>
    <hypernym name="">arrangement</hypernym>
    <hypernym name="">organization</hypernym>
    <hypernym name="">organisation</hypernym>
    <hypernym name="">system</hypernym>
    <hypernym name="">metaphysics</hypernym>
  </hypernyms>
  <hyponyms/>
  <word>concept</word>
  <TopicDomains/>
 ▼<definitions>
  ▼<definition name="">
     an abstract or general idea inferred or derived from specific instances
    </definition>
  </definitions>
 ▼<synonyms>
    <synonym name="">concept</synonym>
    <synonym name="">conception</synonym>
```

Figure 11: Result of the ontology created.



```
<Ontologies>
<word>ontology</word>
<TopicDomains>
 <domain name ="">computer_science</domain>
 <domain name="">computing</domain>
</TopicDomains>
<definitions>
  <definition>The metaphysical study of the nature of being and existence</definition>
  <definition>(computer science) A rigorous and exhaustive organization of some knoweldge domain
   entities and their relations.
 </definition>
</definitions>
<synonyms>
 <synonym>ontology</synonym>
<synonyms>
<hypernyms>
 <hypernym>system</hypernym>
 <hypernym>metaphysics</hypernym>
<hypernyms>
<hyponyms/>
</Ontologies>
```

Figure 12: Format of the ontology created.

In fig.12 the tree structured skeleton of the ontology created is displayed, the diagram is an abstract representation of the structure each ontology that is created follows. The structure is based on the hierarchy of the lexical relation of each word in the ontology, starting with the domain to which the word belongs to , moving to the meaning or the definition of the word, and then to the lexical relations of synonyms, hypernyms and hyponyms. fig.11, in contrast displays how the actual ontology will look like.

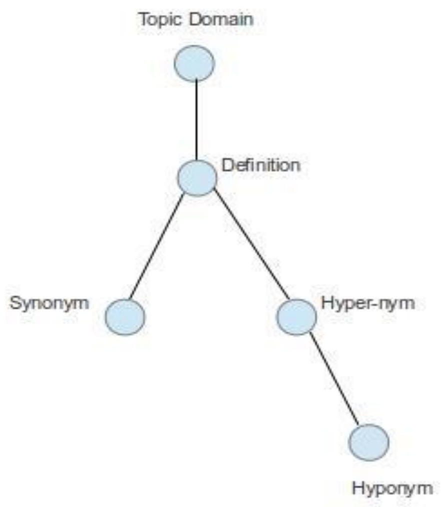


Figure 13: Tree structure of the ontology.

5. CONCLUSION

In this paper an automated method of creating ontologies has been proposed and tested. The system proposed performs the tasks of document classification, searching for the most similar documents, summarizing text and creating the ontology in XML format. Document classification helps to label any new document that is input into the system, tagging the domain to which it belongs to. Document similarity implemented using vector space model is used to find the most similar documents to the input document, from within the data set of documents present in the system, leading to a richer ontology being formed after getting closely related terms to the terms in the input document. Summarizing performs the function of shortening the large text thus formed and keeping mostly relevant, required terms of the document.

The ontologies thus formed are based on lexical relations of a word i.e synonyms, concepts of hypernymy and hyponymy, meaning of the words. In the future applications of such systems can be used in word processing software's, pdf readers, to predict a document the user is reading and get various, accurate details of important terms and the related terms as well.

REFERENCES

[1] Herman I, “Introduction to the Semantic Web”, *Johnson & Johnson,* W3C Consortium, 2009.

[2] Segaran, Toby, “Programming the Semantic Web”, *O'Reilly Publications,* 2009.

[3] Song, Liu, Wang and Wang, “A Novel Automatic Ontology Construction Method based on Web Data”, *Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing,* IEEE Computer Society, 2014.

[4] Hong Sie, S, and Hua Yueh, J. , “Automatic Ontology Creation Using Schema Information”, *Proceedings of the 2006 IEEE/ACM/WIC Conference on Web Intelligence,* IEEE Computer Society, 2006.

[5] Seremeti, L. and Kameas, A, “A Task-Based Ontology Approach for Novice Ontology Developers”, *Fourth Balkan Conference in Informatics,* IEEE Computer Society, 2009.

[6] Hussain, F.K, Sidhu, A.S, Dhillon, T.S and Chang, E. “Engineering Trustworthy Ontologies; Case Study of Protein Ontology”, *Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems,* IEEE Computer Society, 2006.

[7] Jain, Sonal and Pareek, J, “Automatic Topic Identification from Learning Material: An Ontological Approach”, *ICCCEA, 2010.*

[8] “Bayes Theorem”, *Wikipedia the Free Encyclopedia.*

[9] “Naive Bayes Classifier”, *Wikipedia the Free Encyclopedia.*

[10] “Vector Space Model”, *Wikipedia the Free Encyclopedia.*

[11] Russel, M, “Mining the Social Web”, *O'Reilly Publications,* 2013.

AUTHORS

Preeti Kathiria is an Assistant Professor in the Computer Science and Engineering Department at Nirma University in Ahmedabad.

Sukraat Ahluwalia is a final year student in the Computer Science and Engineering Department at Nirma University in Ahmedabad.