CHAPTER 5

ONTOLOGY CONSTRUCTION USING COMPUTATIONAL LINGUISTICS FOR E-LEARNING

5.1 ONTOLOGY CONSTRUCTION TECHNIQUE

Semantic Web (Berners-Lee et al 2001, Buitelaar and Declerck 2003 and Gangemi and Mika 2003) is about managing resources (e.g. Web documents) intelligently especially in heterogeneous systems by describing the properties of the entities (terms) involved and the relationships among them. Such conceptual organization is facilitated by building ontology pertaining to a particular domain (Gómez et al 2003 and Horrocks et al 2003). Domain ontology provides the contents, essential properties in them and the relationships among the terms present in the knowledge base and they have gained enormous popularity in the areas like semantic web, knowledge representation, knowledge management, information retrieval, information search, etc (Maedche and Staab 2000). One of the major areas of research in retrieving the web information intelligently is the provision of learning course contents through online (e-learning) (Fok and Shing Ip 2004). The main motivation to proceed with this work came from several existing educational ontology like EduOnto (An ontology for Educational resources), OntoEdu (Ontology-based Education Grid System for e-Learning), Gene Ontology and OntoGeo. The crux of the online learning course contents in an educational system (e-learning) in the input domain are very well represented using ontology. The main scope of building ontology in this work for the learning course contents (posted by an end-user or an instructor) is to organize them
semantically and such semantic organization of several individual learning
course contents can help in clustering the web documents (learning course
contents) with respect to a particular educational domain.

There are several issues which are yet to be solved in ontology
construction expressively which includes identifying relationships among
terms, hierarchies and mode of visualization (Maedche and Staab 2000).
Traditional ontology building methodologies were text-based models and are
then replaced by graph-based models in the past (Chein and Mugnier 2009).
The existing models in ontology construction helped in representing several
structural relationships among the terms which were depicted clearly, crisply
and in an elegant manner. In the midst of such inventions, identifying and
resolving the presence of anaphora and cataphora among the sentences
pertained to a particular domain was a milestone to be achieved until 1998
(Guangzuo et al 2004, Katja and Malvina 2004 and Webber et al 2003). Most
of the earlier techniques failed when ontology had to be constructed and
visualized for an abstract (web data corpus) with enormous number of related
sentences. The existing methodologies presented in literature survey analysis
lacked the property of resolving anaphora and cataphora and was found to be
a great challenge (Declerck 2002). In our proposed framework DEKA (based
on Computational Linguistics), the construction and visualization of domain
ontology automatically for a large corpus is handled by propositional logic-
based modeling of knowledge base from a computational perspective. DEKA
framework in this work addresses the problem of anaphora resolution
specifically and presents a separate module for resolving intra-sentential and
inter-sentential anaphors.
5.2 DEKA – DECISION ENGINEERING USING KNOWLEDGE APPROACH - THE PROPOSED FRAMEWORK

Computational Linguistics is considered to be a sub-field of Artificial Intelligence and is a field of study devoted to develop algorithms and software for intelligently processing natural language data. The text data present in the natural language should be carefully organized such that information extraction/retrieval could be done intelligently. For the purpose of such semantic organization, a framework for fully automatic domain ontology construction is proposed which is based on Computational Linguistics. DEKA is based on XML Path expressions. The input text is preprocessed and the output is written in an XML file for further processing of the text in ontology construction. DEKA framework constitutes the following modules acting dependently on each other for ontology construction and visualization. The modules are elaborated as

1. Split Sentences
2. Parse Sentences
3. Resolve Anaphora
4. Recovering Ontological Relationships
5. Visualizing Ontology

The complete framework of the system is shown in Figure 5.1.
5.2.1 Split Sentences

The learning course content for an e-learning domain is posted to an e-learning service provider by the end-user (student or instructor). The contents are posted as a text document. This raw document has to be pre-processed before ontology construction. The statements in the text document are split as individual sentences using Sentence Splitter Tagger which analyzes the stop points using the parser (De Marnee and Manning 2010). The output of this module is the individual sentences to be considered for further processing. Each of the individual sentences is given as an input to the next module of Parsing. These stages are considered to be pre-processing stages.
Algorithm : Sentence Splitter

Input : Text document (unprocessed)

Output : Individual sentence

Procedure

Begin

do

    input the text file document to the parser

    document content are split using stop points

while (end of document)

end

5.2.2 Parse Sentences

The document is now ready to be pre-processed completely. The next stage is to provide the document to the parser module. DEKA uses the standard Stanford Parser for parsing the document (De Marnee and Manning 2010). The parser generates the parse tree. This parse tree is given as an input to the typed dependencies generator module of the Stanford Parser. The complete typed dependencies of the document is generated and stored in the XML file. The compound words are identified from this document. Once the compound words are identified the other words are lemmatized to resolve the root words by the process of lemmatization (Miller et al 1990). On successfully finding the compound nouns and root words by the process of lemmatization, the entire document is now cleaned by removing the existing prepositional dependencies like det, auxpass, advmod, advcl, rel, expl,prt, mark. These tags mainly give relation between articles and the words and hence could be removed explicitly. These compound words and the root
words are replaced in the original document. The complete document is again parsed to get the new typed dependencies for further refinement.

**Algorithm** : Parsing

**Input** : Set of statements S in Pre-processed Text Document P, Set of CompoundWord Identification Rules, CWIR

**Input Rule 1** : If a noun is followed by another noun it is a compound noun

**Input Rule 2** : If an adjective is followed by a noun it is a compound noun

**Output** : Set of Cleaned Lexical Semantic representation C

**Procedure**

Begin

CW is set of Compound Words

For each statement $S_i$ in S do

$C_i \leftarrow$ Stanford Parser($S_i$)

For each rule CWIRj in CWIR

$CW_{new} += \text{ApplyRule}(\text{CWIR}_j, C_i)$

$i \leftarrow i + 1$

End

For each statement $S_i$ in S do

For each word $cw_{new}$ in CW

If $cw_{new}$ exists in $S_i$

$S_i \leftarrow \text{Replace}(cw_{old}, cw_{new})$

End

End

End

return $C_i$
5.2.3 Resolve Anaphora

In linguistics, anaphora is an instance of an expression referring to another. The process of anaphora resolution is the problem of resolving what a pronoun or a noun phrase refers to that is previously defined in the complete text document. In other words, the referential entity is called as an anaphor and the entity to which it refers to previously is called as an antecedent. The process of determining the antecedent of the anaphor is called as Anaphora Resolution. Anaphora resolution is a complicated problem in computational linguistics and is an active area of research. The existence of several anaphor in the document has a great impact in the construction of correct ontology. Several types of anaphora are exists which includes pronominal anaphora, definite noun-phrase anaphora and One-anaphora (Denis and Baldridge 2007). DEKA framework consists of a module “Anaphora Resolution” which concentrates on three categories of nouns which pertain to the gender, person and grammar number resolution of the text. The procedure could find he/she/it and who/where types of words for the replacement of nouns.

Algorithm : Anaphora Resolution

Input : Text document (processed - individual sentences)

Output : Resolved Anaphors

Procedure

Begin

Do

{

Provide identifiers for each sentences,  $S_1,S_2,\ldots,S_n$

}
// Anaphora Resolution for he/she/it kind of words

If there exists word $W_i$ in sentence $S_k$ such that Personal-Pronoun ($W_i$) is true then

If there exists word $W_j$ in Sentence $S_{k-1}$ such that Noun ($W_j$) is true then

$W_i$ is anaphora of $W_j$

Else

Display message “Unknown Phrase”

// Anaphora Resolution for who/where kind of words

If there exists word $W_i$ in sentence $S_k$ such that POS ($W_i$) = ”WH” then

If there exists word $W_j$ in Sentence $S_k$ such that Noun ($W_j$) is true and Gender ($W_i$)= Gender($W_j$) then

$W_i$ refers to $W_j$

else

Display message “Unknown Phrase”

}

While (end of document)

End

This algorithm is useful for effective anaphora resolution technique. The time complexity of the proposed algorithm is $O(n)$ where $n$ is the total number of documents considered for anaphora resolution.
5.2.3.1 **KADE – Knowledge Aided Decision Algorithm - proposed algorithm**

The motivation of this enhanced Pronominal Anaphora Resolution algorithm KADE was from the theoretical background provided in the previous work by Shalom Lappin and Herbert Leass (1994), which was an attempt at providing the domain independent anaphora resolver. KADE follows the algorithmic steps similar to the algorithm given by the authors mentioned above with the exception that KADE resolves inter-sentential anaphors. The key power of KADE algorithm is that the existence of related anaphors found anywhere in the web input text corpus or standard corpus could be identified and replaced.

This proposed KADE algorithm is an enhancement of the previous one by providing an additional feature for resolving the anaphors among the different sentences (inter-sentential anaphora detections). Increased efficiency in resolving the anaphors is obtained in this algorithm because the lexical knowledge with respect to a particular domain of the text corpus through Natural Language Processing is considered (Pisanelli et al 2000, Jurisica et al 2004). On performing many empirical tests on various input text corpus, the performance in retrieving the correct anaphors between different sentences (inter-sentential anaphors) is found to be better than many of the traditional works handled. This proposed algorithm KADE however uses the output of Stanford Parser. The overall architecture of the system is shown below
The input to the algorithm is any type of web input text corpus (web search engines) of any length. Initially, all the sentences of the text corpus are provided with an identification number for the purpose of easy referencing. The typed dependencies among the different words in the raw sentences of text corpus are resolved using the traditional Stanford Parser. Many unwanted dependencies may exist when using the parser and such dependencies must be removed. The cleaning process from the typed dependencies obtained earlier, is done by writing specific rules for identifying the compound words, lemmatizing the words and removing the unwanted tags. The compound words in this proposed algorithm is identified by writing the rules like, a noun followed by another noun and a noun prefixed by adverbial modifiers is considered to be a compound noun. Once, the
compound nouns are identified for the entire corpus, the document is cleaned by just deleting the unwanted tags. The anaphors existing in different sentences are identified by allocating an identifier like, CC for Coordinating Conjunction, DET for DETerminer, JJ for adJectives, NN for noun singular, NNS for noun plural, RB for adverb, etc and resolving the POS for each word in the sentence. Such identifier allocation and POS tagging is done using the Penn Tree bank (Marcus et al 1993). On completion of the execution of identifying POS tagging for every word in the sentence, the list of anaphors is displayed. The algorithmic steps of the enhanced anaphora resolution algorithm KADE follow the procedure given below.

**Algorithmic Procedure**

**Premise** : Natural Language Processing  
**Domain** : Text Corpus  
**Input** : Any web input text corpus  
**Output** : List of Anaphors found  

**Procedure**

Begin  

do  
  {  
    // Step 1: Sentence Splitter  
    // Step 2: Resolving typed dependencies among the raw sentences  
    While (end of statement)  
    {  
      Assign Identifier Number for each sentence  
      Describe the grammatical relationships in a sentence among words (nsubj, nn, det, prep, etc)  
    }  
  }  

// Step 3: Compound Nouns Identification using rules description
For (every sentence from the input text corpus)
{
    If a noun followed by another noun then it is a compound noun
    If a noun prefixed by an adverbial modifier then it is a compound noun
}

For (every sentence after identifying compound nouns)
{
    Replace the compound nouns in input text corpus}
}

// Step 4: Resolving Anaphors among the sentences (inter-sentential anaphora detections)

While (end of statement)
{
    Identify the ID allocator and resolve POS using Penn Treebank (Stanford Parser)
}
End; While (end of document)

5.2.3.2 Results and discussions

The basic integrated development environment was developed to test the results for the experimental data sets done by KADE algorithm. The experimental tests were done on several raw input text corpuses. The performance efficiency in terms of correct retrieval of anaphors from the input corpus was found to be an average of 85%. Some of the sample data sets that were taken for the empirical tests for the exact retrieval of anaphors from the text corpus were Doctor Information System, Patient Information System, University Information System, Ontology Information Retrieval, etc (Pisanelli et al 2000). The simplified screen shots for the algorithm evaluation are given below. Screen shots for a very small text corpus is shown in Figure 5.3. The list of anaphors resolved using the input text corpus is shown in Figure 5.4.
5.2.3.3 Sample input text corpus

In this text corpus, every patient has a patient number. This number is used to identify the record of the patient. For every record there is a separate slot to hold the details of doctors who checked the patient and the medicines that they should take.

Figure 5.3 Anaphors Resolution

Figure 5.4 List of Anaphors Resolved
KADE algorithm is evaluated against the traditional performance parameters precision and recall. Precision evaluates the correct number of pronominal anaphors retrieved to the actual pronominal anaphors present in the corpus. Performance parameter recall evaluates the correct number of pronominal anaphors to the guessed pronominal anaphors in the corpus given by the domain expert. Precision and recall values are formulated and shown in Equations (5.1) and (5.2).

**Assumption**

Let \( k \) be the number of actual anaphors present in the text corpus. Let \( c \) be the number of correct anaphors obtained from the text corpus using any anaphora resolution algorithm. Let \( g \) be the number of correct anaphors given by the user, preferably a domain expert.

\[
\text{Precision} = \frac{c}{k} \\
\text{Recall} = \frac{c}{g}
\]  

(5.1) 

(5.2)

The experimental results for different data sets, randomly collected abstract documents from the web engines viz. Doctor Information System (DIS), Patient Information System (PIS) and Ontology Information Retrieval (ORS) are shown in Table 5.1 and their corresponding graphical results are shown in Figures 5.5 and 5.6.
Table 5.1 Evaluation Results – Precision and Recall

<table>
<thead>
<tr>
<th>Text Corpus Files</th>
<th>Number of Actual Anaphors</th>
<th>Precision Value</th>
<th>Recall Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doctor Information System (DIS)</td>
<td>77</td>
<td>0.7</td>
<td>0.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.77</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.7</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>Patient Information System (PIS)</td>
<td>57</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.7</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.68</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>Ontology Retrieval System (ORS)</td>
<td>130</td>
<td>0.82</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.85</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.96</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5.5 Precision Comparison
From this analysis, it can be observed that the precision and recall for the proposed KADE algorithm is better when it is compared with the existing algorithms namely Java RAP and Hobb. This is due to the fact that this proposed algorithm resolves not only intra-sentence anaphors but also inter-sentence anaphors.

### 5.2.4 Recovering Ontological Relationships in DEKA

The output document obtained from the previous stage is present as lexical semantic representation. The exact hierarchy for the ontology construction is designed in this phase. The relationships among the terms are to be resolved for the generation of ontology. The first-order logic implementation is used in order to write the facts and rules to be applied on the document. From the typed dependencies, the prolog facts are obtained. The nouns and verbs from the documents are identified in order to find the relationships using rule-based approaches. The basic facts and rules are applied on these prolog facts. The query processor runs and finds the different kinds of structural relationships among the terms in the document. The results
obtained are cleaned and the following six types of structural relationships are obtained (Assawamekin et al 2009). They are categorized as

1. Aggregation relationship (partOf)
2. Attribute relationship (attributeOf)
3. Property relationship (propertyOf)
4. Generalization/specialization relationship (isA)
5. Equivalence relationship (sameAs)
6. Association relationship (associatedWith)

Once these structural relationships are identified, the connectives among the terms could be easily resolved and viewed (Assawamekin et al 2009). The algorithmic steps are given below

**Algorithm** : Ontological Relationships

**Input** : XML file with typed dependencies

**Output** : Prolog file with facts and generated ontological relationships

**Procedure**

Begin

Obtain prolog facts from the typed dependencies

Knowledge base (facts and rules) is written using first-order logic extraction rules

Query processor identifies the six kinds of relationships

End
Extraction of Rules

Rule 1 and 2 - Object and Relationship: If ‘x’ is a noun, then ‘x’ is an object. If ‘r’ is a verb, then ‘r’ is a relationship.

Rule 3 - Aggregation: Let ‘r’ be a special verb relation (e.g., part of, belong to or subdivision of). If a noun ‘x’ participates in object ‘y’ with ‘r’, then object ‘x’ is apart of object ‘y’.

Rule 4 - Aggregation: Let ‘r’ be a special verb relation (e.g., have, contain, comprise, include, define, consist of, compose of, denote by, identify by, make up of or record with). If a noun ‘x’ participates in a noun ‘y’ with ‘r’ and a noun ‘y’ also participates in a noun ‘z’ with another ‘r’, then noun ‘y’ is a part of noun ‘x’.

Rule 5 - Attribute: If a noun ‘x’ participates in a possessive relation with another noun ‘y’ and is mentioned once in the requirements artifacts, then the noun ‘x’ is an attribute of the noun ‘y’.

Rule 6 – Generalization/Specialization: Let ‘r’ be a special verb relation (e.g., be, kind of, type of, classify into or consider as). If a noun ‘x’ participates in noun ‘y’ with ‘r’, then the noun ‘x’ is a kind of the noun ‘y’.

Rule 7 - Association: Let ‘r’ be a relationship. If a noun ‘x’ participates in the noun ‘y’ with ‘r’, then the noun ‘x’ is associated with the noun ‘y’ with relationship ‘r’.

Rule 8 - Property: If a noun ‘x’ is a numeric modifier of another noun ‘y’, then the noun ‘x’ is a property of the noun ‘y’.

Rule 9 - Equivalence: If a noun ‘x’ is an abbreviation for another noun ‘y’, then the noun ‘x’ is the same as another noun ‘y’.
5.2.5 Visualizing Ontology

The structural relationships that are obtained are made feasible through any visualization mechanism. For such visualization, a base ontology (Assawamekin et al 2009) is used as a premise. The generated relationships are matched with the base ontology and create a new ontology graph. The base ontology is given as a graphViz dot file. The new relationships are matched with the base ontology and written back to graphViz dot file. The new ontology graph is now visualized using graphViz tool.

Algorithm : Ontology Creation

Input : Relationship file, Base ontology file

Output : Ontology graph

Procedure

Begin

Base ontology is given as graphViz dot file

Ontological relationships file are mapped with the base ontology mapper

New ontology is visualized using graphViz tool

End

5.3 EVALUATION RESULTS

The proposed DEKA framework encapsulates various implementation language and standard tools for performance evaluation. The implementation languages and tools used in our approach are java, Prolog, graphViz dot, Netbeans IDE 6.9.1, Stanford Parser, tuprolog and graphViz. The input text is taken from the web corpus. The abstracts posted by the
end-user or the instructor to the e-learning service providers are used for the evaluation of the performance metrics. The performance efficiency in terms of constructing the right ontology by our algorithm is found to be more than 85% (approx). The sample input data set considered for our approach is University Information System, School Educational System and Library Information System (Assawamekin et al 2009).

5.3.1 Evaluation Procedure

When evaluation and testing had to be performed, we obtained the help of some of the domain experts in identifying the ontological relationships. They worked out manually with the help of a paper and pen for the construction of ontology and recovering ontological relationships. Later, the same web document input was given to the several algorithms present in the system. The number of ontological relationships uncovered by both the domain expert and the system were stored for analyzing the performance metrics. DEKA framework is evaluated using a number of web document abstracts. The framework is tested against the traditional performance metrics, precision and recall. It was found that good results were obtained in comparison to the other algorithms explained in the literature survey.

5.3.2 Results Analysis

The values of the performance metrics - Precision and Recall achieve better results when the size of the dataset is less. As the size of the data set increases the values also tend to be low. Hence, the main limitation of our framework is that the algorithm sometimes fails to achieve scalability. In connection to this, the current ongoing work aims at solving the problem of scalability. The graphical results for precision, recall and the performance
efficiency is shown below. As a way to this, the formulation for precision, recall and performance efficiency are given below

**Assumptions**

Let m (integer variable) be the number of ontological relationships uncovered by a domain expert.

Let n (integer variable) be the number of ontological relationships uncovered by the system (involving various algorithms).

Let c (integer variable) be the total intersecting number of ontological relationships uncovered by both the domain expert and the system.

Let g (integer variable) be the number of exact (correct) number of ontological relationships present in the web corpus.

Let P, R, E be the variables for precision, recall and performance efficiency output values.

The value of precision is given by,

\[ P = \frac{c}{n} \]  \hspace{1cm} (5.3)

The value of recall is given by,

\[ R = \frac{c}{m} \]  \hspace{1cm} (5.4)

The performance efficiency is calculated as,

\[ E = \frac{c}{g} \]  \hspace{1cm} (5.5)
The precision, recall and performance efficiency values obtained for the proposed DEKA were compared with the existing algorithms namely TextOntoEx, OntoLT and Text-To-Onto and are shown in Tables 5.2 to 5.4. They are also shown graphically in Figures 5.7 to 5.9.

### Table 5.2 Precision Evaluation

<table>
<thead>
<tr>
<th>Text Corpus</th>
<th>Algorithms</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TextOntoEx</td>
<td>OntoLT</td>
<td>Text-To-Onto</td>
<td>DEKA</td>
<td></td>
</tr>
<tr>
<td>University Info System (1)</td>
<td>0.96</td>
<td>0.90</td>
<td>0.86</td>
<td>0.96</td>
<td></td>
</tr>
<tr>
<td>School Educational System (2)</td>
<td>0.90</td>
<td>0.88</td>
<td>0.83</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>Library Information System (3)</td>
<td>0.91</td>
<td>0.86</td>
<td>0.81</td>
<td>0.94</td>
<td></td>
</tr>
</tbody>
</table>

### Table 5.3 Recall Evaluation

<table>
<thead>
<tr>
<th>Text Corpus</th>
<th>Algorithms</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TextOntoEx</td>
<td>OntoLT</td>
<td>Text-To-Onto</td>
<td>DEKA</td>
<td></td>
</tr>
<tr>
<td>University Info System (1)</td>
<td>0.89</td>
<td>0.87</td>
<td>0.84</td>
<td>0.88</td>
<td></td>
</tr>
<tr>
<td>School Educational System (2)</td>
<td>0.87</td>
<td>0.85</td>
<td>0.83</td>
<td>0.86</td>
<td></td>
</tr>
<tr>
<td>Library Information System (3)</td>
<td>0.81</td>
<td>0.78</td>
<td>0.75</td>
<td>0.81</td>
<td></td>
</tr>
</tbody>
</table>
Table 5.4 Performance Efficiency

<table>
<thead>
<tr>
<th>Text Corpus</th>
<th>Performance Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TextOntoEx</td>
</tr>
<tr>
<td>University Info System (1)</td>
<td>0.97</td>
</tr>
<tr>
<td>School Educational System (2)</td>
<td>0.95</td>
</tr>
<tr>
<td>Library Information System (3)</td>
<td>0.9</td>
</tr>
</tbody>
</table>

The graphical visualizations of the performance metrics namely precision, recall and performance efficiency are shown in Figures 5.7 to 5.9 respectively.

**Precision**

![Figure 5.7 Precision Evaluation](image)
Recall

Figure 5.8 Recall Evaluation

Performance Efficiency

Figure 5.9 Efficiency Evaluation
From the tables and graphs it can be observed that the precision, recall and performance efficiency of the proposed DEKA framework model is better when it is compared with the other existing models. This work used ontology for evaluation since ontology plays a vital role in clustering the web documents semantically to enhance the performance of many information extraction and information retrieval systems. This work has addressed the sensitive issue of resolving anaphors. The main advantage of this work is that a complete framework has been designed for parsing, anaphora resolution, uncovering ontological relationships and visualizing the ontology.