CHAPTER 1
INTRODUCTION

1.1 GENERAL

Human has assembled information into the greatest repository called World Wide Web having information resources even on unthinkable subjects. This information may be available instantly to anyone having Internet connection. As more and more people are learning its usage, they can add more content on the web. As a result of this web is growing exponentially and it is becoming difficult to locate useful information in such a sheer volume of information. Sometimes finding the relevant and correct information is like finding a needle in haystack. Moreover there may be requirement of collecting information from a series of web pages and integrate that information or perform some reasoning on that information. For making this procedure faster, the machine must understand the text and should be able to process that text.

1.2 SEMANTIC WEB

There is a proposal from Tim Berners-Lee to augment the existing web with information which makes the meaning of web pages explicit. He has devised the term Semantic Web and according to him “The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [1]. Whereas web 1.0 and web 2.0 contain minimal machine-processable information in the dumb links, Semantic Web is web 3.0 technology which is enhancement of the current web with more machine-processable information. It is a way of linking data between systems or entities that allows for rich, self-describing interrelations of data. Semantic Web has opened up the web of data to artificial intelligence processes. Figure 1.1 shows semantic web extending the current web by emphasizing on interoperable ontologies which are capable of processing high quality information so that the agents placed on top of semantic web can automate the work or curate the content for the user.

1.3 NEED AND APPLICATIONS OF SEMANTIC WEB

Though many search engines are capable of indexing significant portions of the web, there are some major problems for users such as getting so many irrelevant results or getting no result at all. The first problem occurs because a word may have different meaning in different contexts, Moreover the search engines index information using keyword indices that do not preserve any relationship between words. Another problem may be that the searched term or terms may not appear on web. There is also possibility that the desired document is missed by search engine.

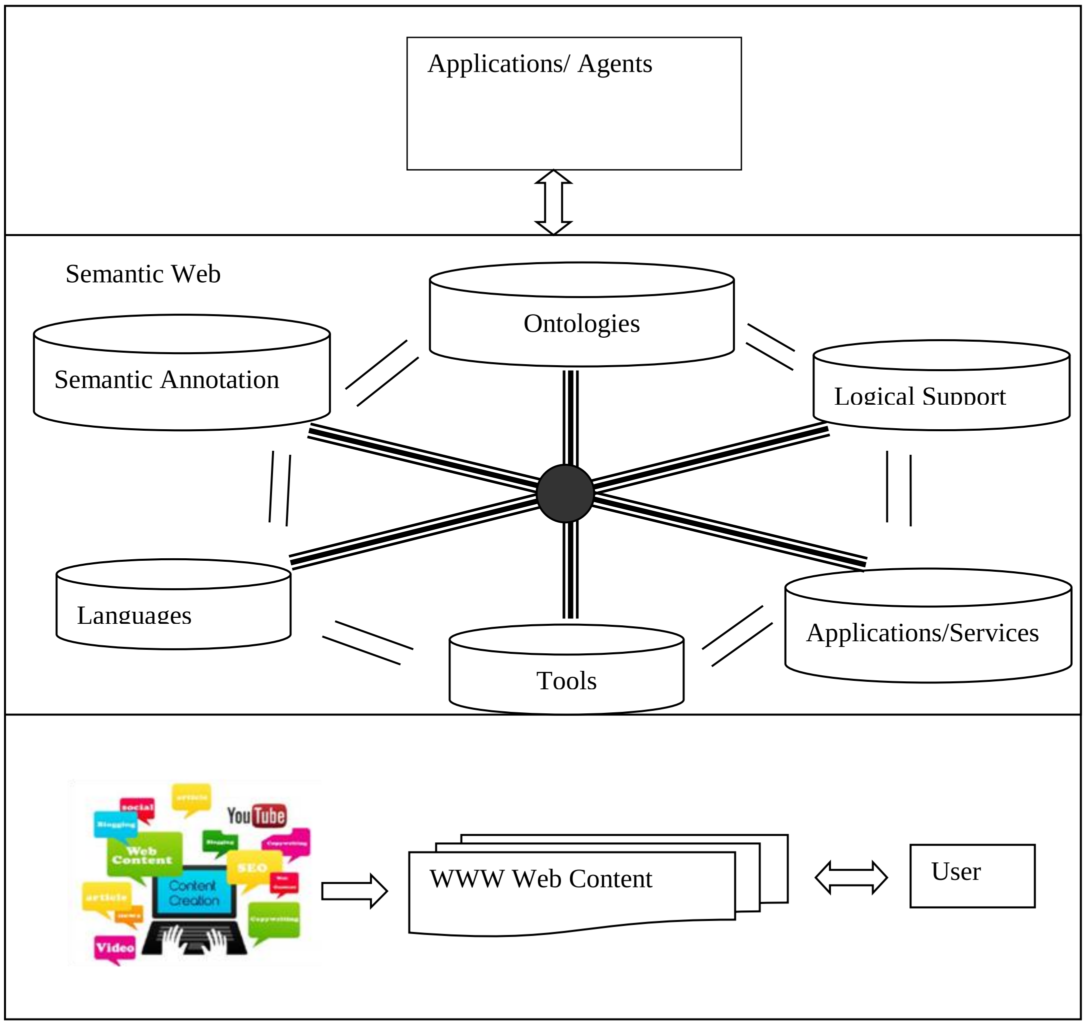


Figure 1.1: Semantic Web

This lack of ability of search engines to understand the context of words and relationships between search terms may identify many false positives. If there is any likelihood for a search engine to understand the intended meaning of the words or find whether there are some semantic relationships between them, they will be capable of providing more accurate searches. This is one of the aims of the Semantic Web.

Semantic Web has its major use in Knowledge Management, ease of information integration, more efficient searching, more effective reuse of information etc.

1.4 KNOWLEDGE REPRESENTATION IN THE SEMANTIC WEB

Knowledge representation is the formalisation of knowledge and how it is processed by machines. The goal of Semantic Web is to make information ‘understandable’ or ‘knowledgeable’ in true sense for multiple applications through semantic interoperability along with technological interoperability. This requires common understanding and good communication among them by resolving their difference of languages, different structures and methods and having non-overlapping and well mapped concepts or terms which is achieved by analysing the contents and providing the contexts to the information. This leads to using models which provide these specifications explicitly and in some formal way defining the terms and how they are related to each other. These models provide shared and common understanding of a domain that can be communicated across applications and people.

For achieving technological interoperability Extensible Mark up Language (XML) is used. Semantic interoperability can be achieved using richer language such as RDF, OWL etc. These models can be used for inference and information exchange [2].

XML-XML is used as a simple way to transfer documents across the Web. Anyone can design own document format and then write some document in that format. These formats can include “machine-readable” markup also to enhance the meaning of its content. By including these mark-ups in the documents, they become more powerful.

Resource Description Framework (RDF) - It is a format for defining the information on the web. RDF is also a markup language like XML but is used for describing the information and resources on the web. It makes easy for the web agents to search and process information from the web which has been put into rdf files. RDF describe web resources and provide a syntax such that individual parties can use and exchange information. It is not designed for being represented to human. Its motive is to be machine readable as well as machine understandable.

RDF Schema - RDF Schema is a datatyping model for RDF. Using RDF Schema, it can be said that "Tom" is a "Cat", and that "Cat" is a sub class of animal. It can be used to create properties and classes and can provide ranges and domains for the properties. The most important concepts given by RDF and RDF Schema are the "Resource" (rdfs:Resource), the "Class" (rdfs:Class), and the "Property" (rdf:Property).

Ontology - Ontologies are considered powerful tools for simulating the conceptual models because of their expressiveness, effective knowledge representation formalism and associated inference mechanisms [3], [4]. Now a days, ontology is used for knowledge representation in information retrieval, artificial intelligence and semantic web. Ontology has a great use in the web documents retrieval as keyword-based searching may retrieve information with some false matched results because there is only information retrieval by matching keywords and not extraction by matching the meaning of the keywords can be done by using ontology.

Ontology is a structure made up of categories of objects or ideas in the world, along with certain relationships among them. In general, ontologies are knowledge structures that adopt a rich formal language aiming at classifying notions of interests like process, event, quality, object and so on. The task of representation of Ontology is a way of standardizing information for more flexibility, and to enable more rapid development of applications, and sharing of information.

Following aspects are described by ontology as given by [5]:

Concepts:- A concept may represent a group of different individual objects in the mind sharing the same characteristics. These may also be called classes which are abstraction of objects. Concepts may be concrete (like people, countries), abstract (like fact, goal, belief), fundamental or complex. Generally these concept or classes are a part of hierarchical organisation to reduce the space requirements and are able to follow inheritance mechanism.

Instances:- These are the individuals or things that are described by ontology and elements belonging to a specific class. These can be actual objects such as person's name, a location, a person's car etc. or an abstract entity. These are the base unit of ontology.

Relations:- The way in which individual classes or instances relate to each other is described by the relations. Relation may be considered as an attribute, the value of which is another object of the ontology. The whole semantic in a domain is described by these inter-related relations.

Attributes:- Specific information of an object is stored in the form of attributes. Ontology describes its objects by assigning at least one name and value to its attributes.

Ontology Classification: Ontology is categorized according to the level of generality as follows:

● Top-level ontology: This ontology is concerned with general concepts e.g. time, event, action, matter, etc.

● Domain ontology: A common vocabulary is provided to a domain by this ontology, so various domain knowledge can be understood and exchanged. Domain ontology examples are: Music ontology, Geo ontology, Food ontology, Gene ontology, etc.

● Application ontology: These ontologies are built for some specific purpose to share knowledge modeling among various domains.

- Task ontology: This ontology is based on task or activity e.g. buying or diagnosing.

1.5 ONTOLOGY LEARNING AND CHALLENGES INVOLVED

The task of automatic or semi-automatic construction of ontologies from domain data is called Ontology Learning [6]. A general process of ontology learning is shown in Figure 1.2, given as follows:

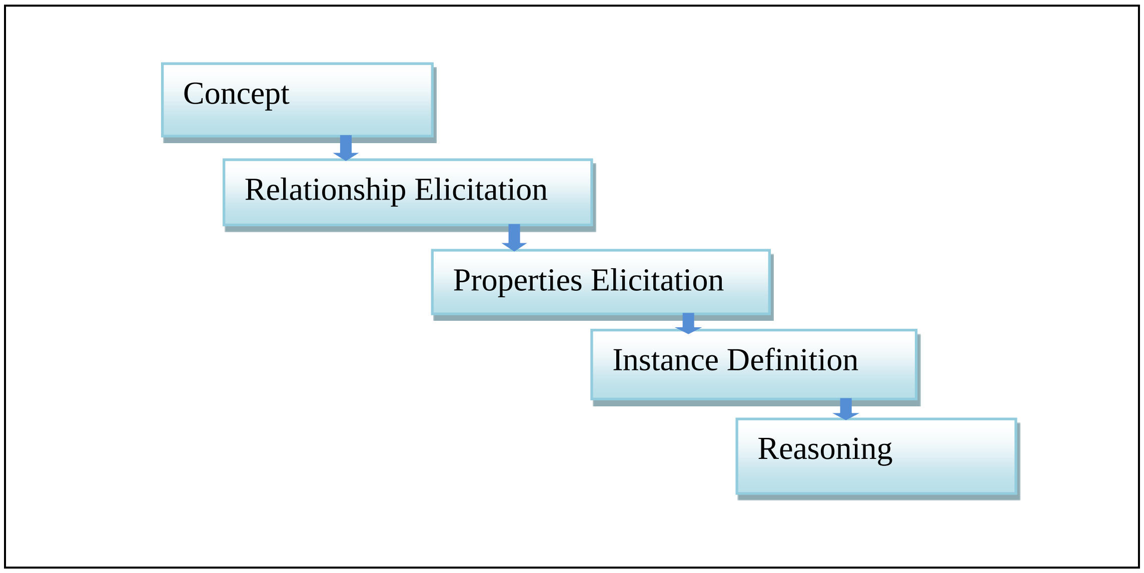


Figure 1.2: Ontology Learning

However, few other researchers have established their own methodologies for ontology learning.

UPON-Lite [7] methodology includes six steps of ontology learning:

a) listing of domain terminology

b) lexicon terms associated with their synonyms

c) is-a hierarchy generation

d) predication

e) finding meronymy

f) coding ontology formally

GOSPL [8] has aimed at building hybrid ontology where concepts are defined both formally and informally. This methodology provides collaboration between the ontology engineers and domain experts and uses Glossary as a special linguistic resource. It includes the following steps:

a) Semantic Interoperability Requirements are defined
b) Glossary is built
c) Lexons are created
d) Constraints are put over Lexons
e) Hybrid Ontology is generated
f) Co-evolution of Community and Semantic Interoperability Requirements

NeOn [9]suggests multiple ways of building Ontology instead of giving any one methodology for different situations by re-engineering and reusing knowledge resources:

Situation 1: From specification to implementation- Without reusing the available knowledge resources, ontology is generated from scratch

Situation 2: Reusing and re-engineering non-ontological resources -Ontology engineers analyze non-ontological resources for their reuse according to the ontology requirements.

Situation 3: Reusing ontological resources –ontologies, ontology modules are used for building ontology by ontology engineers.

Situation 4: Reusing and re-engineering ontological resources- ontology engineers both reuse and re-engineer ontological resources.

Situation 5: Reusing and merging ontological resources- Here several ontological resources from the same domain are picked up for reuse.

Situation 6: Reusing, merging, and re-engineering ontological resources- in this situation ontology engineers decide to re-engineer the available resources and merge them.

Situation 7: Reusing ontology design patterns (ODPs) - These repositories are reused by ontology engineers

Situation 8: Restructuring ontological resources- Different restructuring techniques such as extending, pruning or modularizing the different ontological resources is used for building ontology

Situation 9: Localizing ontological resource- one ontology is adapted in different cultures, language or community.

However ontology learning or ontology generation is not an easy task. Following are few important challenges as observed in the process of learning ontology which are faced by researchers given as:

- **Unstructured texts:** There is an open challenge according to S. Gillani Andleeb [10] to learn an effective personalised ontology from the critical information. This information may be scattered amongst various kinds of documents originating from various sources such as emails and web pages or user's local information repository that does not have meta data. For this reason, the results for discovery of relations between concepts are also not satisfactory [11].
- **Ambiguity in English text/Multiple senses of a word:** As there may be multiple senses of word each of these having a different meaning based on the context of the word's usage in a sentence, it has to be resolved. Producing inconsistent or duplicate entries and dealing with these inconsistencies is quite challenging. [12]

Due to above inherent problems the ontology techniques earlier proposed have some deficiencies stated as follows:

i. **Lack of fully automatic techniques for Ontology Development:** Due to unstructuredness and ambiguity in texts, there are very less standard tools for developing ontology as explained by [13] [14]. Existing techniques use supervised learning which require large amount of data for training.

ii. **Ignoring other useful information from the texts:** Existing techniques typically consider word frequencies, co-occurrence statistics, and syntactic patterns such as Hearst patterns [15] and cover only those terms or sentences, (by ignoring others) that satisfy these constraints. The ignored text may also contain useful information such as non-taxonomical relations (relations other than *is-a*, *has-a* or *part-of* etc.) or data properties. In general, the information in a text has multiple layers such as semantic roles denoting the context of a concept and semantic relations. Ideally all levels of information should be used to construct the ontology for a given text.

1.6 PROBLEM DEFINITION

After having a critical look over the work done in this field and considering the limitation of each, it has been observed that constructing ontology automatically is a challenging and important task. The work done till now in automated engineering of ontologies capture only taxonomical relationships such as is-a, part of etc.

Moreover only limited language constructs such as nouns are considered as the building blocks for ontologies ignoring other constituents (such as verbs and adverbs) of a language in the given text. Also larger texts and compound or complex sentence structure in the text imposes difficulty in exploring the semantic content of the text and constructing ontology. In this work an ontology generation technique will be devised covering all important aspects missing in the existing works. Particularly semantic roles along with the other constituents such as nouns, verbs etc. will be used to build ontology without limitation of its size.

1.7 OBJECTIVES OF THE PROPOSED WORK

To gain solution to the problem of developing ontology generating technique, considering the limitations of works done previously in this direction, we formulate the following research objectives:

- **Designing a framework for constructing an ontology**

There is a need for generating an ontology automatically eliciting taxonomic and non-taxonomic relations from an unstructured and semi-structured document which is a tedious task.

Solution: For generating ontology i) from unstructured documents ii) from sources that have some predefined structure, such as HTML a framework is designed which first pre-processes the document to extract semantic roles of nouns in the sentence along with usual concepts and their relationships. The extracted information about different roles, concepts and relationships among the concepts from different documents are then merged to construct ontology for whole document.

- **Enriching the Ontology with classification of data properties**

Ontology gives annotations in the form of rdf or rdfs which are used for providing intelligent services like information retrieval, question answering etc. These services can perform better if ontologies being used contain extra information about the concepts. This information gives an idea about the context of the concept. While extracting ontology from unstructured text ontology can be extended with specific knowledge to provide more information.

Solution: We have defined several classes the data properties of the ontology may belong to. These class labels are stored along with the data properties to enrich the ontology.

- **Refining the Ontology by removing the un-necessary information and summarizing the documents**

Summarization of text is a necessity as there is a large amount of data on the web expressing the same ideas. It requires deciding which sentences or phrases are to be chosen such that they show the main ideas in the document. Even in the sentences chosen there may be redundant and un-necessary information.

Solution: Information which is redundant and unnecessary in the documents is processed to be removed from this document by deleting the un-necessary transition words and then summarizing the documents by using machine learning technique Support Vector Machine (SVM) along with constructing the ontology for SVM processed document and further removing any redundancy in the text.

1.8 ORGANIZATION OF THESIS

The thesis has been organized in the following chapters:

Chapter 1: This chapter starts with brief introduction of semantic web, ontology, challenges faced while learning ontology and also defines the problem which is tried to be resolved in our work by giving the objectives of our work.

Chapter 2: This chapter contains the detailed literature survey done on ontology development techniques considering their strengths and weaknesses. Comparison of different techniques is also provided in the form of a table. This section also includes a brief introduction to the tools and knowledge structure used in our work. The problem definition is also revisited in the light of surveys conducted above.

Chapter 3: In this chapter a novel architecture for automatic construction of ontological framework using conceptualization and semantic roles has been designed and proposed. The architecture depicts different functional modules that are proposed and discussed in this chapter. This chapter also provides implementation details and the result analysis of the experiment conducted.

Chapter 4: This chapter introduces a new technique for enriching the ontology by labeling the data properties extracted by the name of the class of data property to which it belongs. This chapter also provides implementation details of this technique.

Chapter 5: This chapter proposes a new hybrid technique for summarizing the input text by first extracting some statistical features from the text followed by SVM classifier to generate extractive summary. This extractive summary is used to construct ontology using our proposed approach which later is used to generate the final summary of the input document. The final summary (an abstractive summary) generating step involves rewording or reconstructing sentences from the ontology. This chapter also provides implementation details of the system and the result analysis of the experimentation.

Chapter 6: This chapter gives a conclusion about the research work embodied in this thesis and provides insights for extending this work in future.

CHAPTER 2
LITERATURE REVIEW

2.1 INTRODUCTION

This chapter presents a detailed survey of literature related to the proposed work. However, we will provide significant definitions and terminology related to semantic web in section 2.2.

As this thesis deals with providing an ontological framework design for semantic web, section 2.3 dwells on how the methodological and theoretical contributions address the ontology building activities. This review frames the main goal of thesis to build ontology with unstructured text.

We also propose a technique by which ontology can be extended with specific knowledge to provide more information about the constructs of ontology. Section 2.4 explores the approaches to enrich the ontology.

The approach purposed to build ontology from unstructured text in this thesis is utilized to summarize text documents. Next section 2.5 offers insights of some works on text summarization. This section reviews the methodological works and approaches to address this activity.

Section 2.6 reviews the tools used in our work. This section also focuses on the knowledge bases utilized in our work.

Finally, section 2.7 ends the chapter with a conclusion on the analysis of reviewed works and summarises the problems encountered in these works.

2.2 SEMANTIC WEB LAYERED ARCHITECTURE

While the earlier web is used to display the contents on a page using HTTP, the Semantic Web is trying to induce machine readability of content by semantically representing data or information resources. The semantic web addresses the shortcomings of earlier web using the descriptive technologies like Resource Description Framework (RDF) and Web Ontology Language (OWL) and customizable Extensible Mark-up Language (XML). These technologies are combined so as to provide descriptions that support or take the place of the content of Web documents. There are several formats and languages that form the building blocks of the semantic web as shown by the layered architecture of semantic web in Figure 2.1. These include the following components which provide a formal description of concept and relationships among them within a given knowledge domain.

1. Unique Identification Mechanism

Identifiers are used to identify things on the Web. A uniform system of identifiers is used where each thing identified is a "resource". These identifiers are called "Uniform Resource Identifiers" or URIs.

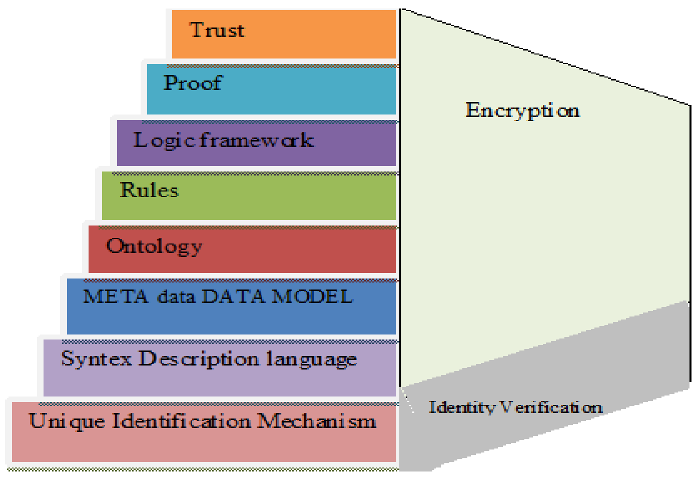


Figure 2.1: Semantic Web Layered Architecture

2. Syntax Description Language

This layer is instantiated by XML that is designed to send and receive documents around the web. It lets anyone to frame their own document format and write a document in the same format. These formats include “machine-readable” markup for enhancing the meaning of the document's content which also makes the documents more powerful.

3. Meta-data Data Model

RDF instantiates the meta-data data model. For achieving easy integration of highly distributed data of Semantic Web, the encoding scheme for description about resources is provided through Resource Description Framework. This is a graph structured data format in which things are denoted by resources that can be a concept or instance from a domain. A special kind of resources called predicate describes the relationships among other resources. These resources be it concepts, instances or predicates are assigned Universal Resource Identifier (URI) and are represented as a set of triples known as rdf statements. Here each triple contains a subject, predicate, and object in the form of <subject, predicate, object> (e.g. <person1, ownerOf, car1>). The subject is the source of an edge, the object is its target and the predicate is the edge itself. Resources which are subjects in some triples may appear as an object in other triples. An inverse predicate (e.g. hasCar) of the predicate (e.g. ownerOf), can exchange the subject and object of a triple. Unlike subjects and predicates which are always resources in a triplet, objects can either be a resource or it can be a literal. A literal could be a string, a number, a date, or some arbitrary sequence of characters. To define the same literal in different languages, a literal could also be given a language tag.

4. Ontology

Ontologies provide knowledge sharing and facilitate reusable web contents and web services as poor communication or lack of common understanding which occurs in web based applications due to mismatched or overlapping concepts or due to different languages may lead to poor knowledge or software interoperability. Ontologies were majorly introduced to solve the problem of poor communication. Ontologies are described by several languages. Some of the ontology languages are OIL (Ontology Interference Layer), DAML (DARPA Agent Markup Language), and OWL (Web Ontology Language). Description Logic and DAML+OIL combined form OWL which is a set of XML elements and attributes having well-defined meaning, which is used to define terms and their relationships.

5. Rules and Logic Framework

For expressing complex mappings among ontologies rule languages are used. Several rule languages are proposed on the top of RDF for example Semantic Web Rule language (SWRL), WRL etc. These languages may offer support for non-monotonic negation or provide rich sets of built-in functions. A generic Rule Interchange Format (RIF) has been standardised by W3C working group in 2005 which now has reached a proposed recommendation status.

6. Proof and Trust

Logic statements written by people in the form of semantic links can be followed by machines to construct proofs. The trust part of semantic web is yet to be implemented. Digital signatures can be used here and own levels of trust can be set so that the computer can make decisions about what to believe and what to distrust.

7. Encryption

A secure version of HTTP known as HTTPs is developed for making sure that data remains unaltered during transmission. HTTPS uses a different server port and an encryption protocol to avoid the man-in-the-middle attacks and eavesdropping on the level of the transport layer. RDF graphs are digitally signed to ensure the authenticity of content and the source that the content has been the same. RDF document signing consists of using standard digital signature methods.

8. Identity Verification

For a decentralized system the issue of identity is important in the context of the Semantic Web as people may use their own identifiers for resources leading to an excess of identifiers for the same entity. Using instance URIs or class URIs already being used by other sources may help in creating a web where distributed discovery is possible. To be able to gather information from different sources the identity of URIs has to be established by performing a syntactic check to merge data when same URI are attached RDF descriptions by two sources.

In the process of devising an effective knowledge representation system for Semantic Web we need to design an Ontological framework for the same for following reasons:

i) For sharing common understanding of information among people or software agents.

ii) Enabling reuse of domain knowledge.

iii) Separating domain knowledge from operational knowledge

iv) Analyzing domain knowledge.

2.3 ONTOLOGY DEVELOPMENT METHODS

To reduce the high cost of building ontologies manually, automatic construction of ontology has been the focus of recent research. [16] [12] [10]

Ontology building approaches can be classified according to the type of knowledge resources i.e. fully structured text such as databases, dictionaries or existing ontologies, semi-structured text such as HTML or XML or unstructured text such as plain text in journals, books ,web.

This section includes a brief review of some approaches to generate ontology from structured and semi-structured text. As the main issue addressed in this thesis is to generate ontology from unstructured text, a detailed description of methodologies and approaches to address this activity is given, in the sub-section 2.3.3, in chronological order and a table summarizing these approaches along with the problems found in each approach is also presented in the same sub-section.

2.3.1 Ontology from Structured Text

Constructing ontology from fully structured text involves updating the existing ontology, extracting ontologies from existing knowledge bases or merging the existing ontologies to build a large ontology. A few approaches have been investigated for developing ontology from existing ontologies.

Hairth Alani [17] gives an approach for constructing ontologies from existing ontologies automatically using ontology mapping and merging techniques. This approach is less costly as it doesn't start from scratch. This approach ranks the ontologies from a domain to get top ranked ontologies. These ontologies are then segmented and merged to form a detailed ontology. A number of technologies are intended to be used such as ontology searching, segmentation, ranking, matching, merging and evaluating the ontology. Problem in this system is that it cannot guarantee the retention of quality and consistency of the original ontologies in the extracted segments which are processed further to be joined as a big ontology.

Junli Li et al. [18] presents a technique to construct geo-ontologies by merging ontologies from various sources. The formal semantics are extracted using Formal Concept Analysis. Information entropy along with deviance analysis is used as a basis for reducing the size of merged concept lattice as preferred by the user. A threshold can be applied for reducing the merged concept lattice in accordance with the user interest. The difficulty with this method is that human intervention is largely expected to maintain the accuracy of merged ontology.

C. P. Abinaya et al. [19] utilize semantic and syntactic measures for merging and identifying similar concepts. WordNet is used here for determining similarities among classes and instances from different ontologies and then merging ontologies. The difficulty with this method is that merging can be done only for the same domain ontologies.

2.3.2 Ontology from Semi-Structured Text

Semi-structured text means data organized into a database so constructing ontology from semi-structured text involves transforming a database into ontology. A few researchers have worked on constructing ontology from database. Overview of some works is given as follows:

Andreia et al. [20] has introduced a transformation algorithm from a database into ontology. In Object-oriented databases have been used to be transformed by this algorithm. The main characteristics of such database types have been considered.

Guntars Bumans [21] shows relational databases can be processed to define a bridging mechanism between relational data and OWL ontology using SQL to generate RDF triples for OWL class and property instances. This technology provides the means to use relational database as a powerful tool to transform relational data to Semantic Web layer.

Kgotatso Desmond [22] proposes two Protégé plug-ins DataMaster and OntoBase that are used to construct ontologies automatically from an Oracle relational database. In addition to this two visualization plug-ins including OntoGraf and OWLViz are used to analyse the semantic structures of the resulting ontologies. One more tool is used as well as an ontology documentation software, namely, Parrot. The performances of the plug-ins were further measured based on the database-to-ontology mapping rules/principles. The results revealed that both tools reasonably convert a relational database to ontology with slight deviations from the database-to-ontology mapping principles.

2.3.3 Ontology from Unstructured Text

Constructing ontology from unstructured text is the process of identifying concepts, relations among these concepts and properties of concepts from textual information and using them to construct and maintain ontology.

Few approaches by some researchers attempt to build ontology from unstructured text manually, semi-automatically or automatically by employing different ideas. We describe here some of the approaches as follows:

i) Statistical Techniques And Hearst's Patterns

Khurshid Ahmad et al. [14] have worked on unstructured text to construct ontology which makes automatic identification of keywords used as concepts using statistical techniques and then using Hearst's patterns [23] to enhance the ontology. In this work domain expertise is needed to provide evidences.

ii) Supervised Approach

A supervised approach to automatic Ontology Population is given by **Hristo Tanev et al.** [24]. They have populated ontology of Named Entities in which geographical locations and person names are used as two high level categories and each category has ten sub-classes. A syntactic model is learnt for each sub-class, using a list of training examples and given a syntactically parsed corpus. Unknown named Entities from the test set are classified using this model. As no annotated corpus is used in the learning process, this approach is weakly supervised.

iii) Concept-Relation-Concept Tuple-Based Ontology Learning

Abbreviated as **CRCTOL** [25] is an approach devised by **Tan et al.** for constructing ontologies from domain-specific documents. For performing ontology learning tasks it uses linguistics and statistics-based techniques. Documents of different formats such as

PDF, XML etc are converted into plain texts using data importer. For part of speech tagging and other syntactic information Stanford's part-of-speech tagger is used along with the Berkeley Parser. Nouns and noun phrases are extracted in the form of multi-word terms using some predefined rules. Terms are identified whether they belong to specific domain using a manually built domain lexicon. Extracted terms are further cleansed by removing adjectives and articles associated with them. Each extracted term is weighed by using Domain Relevance Measure. To find is-a relations lexico-syntactic patterns are used.

iv) Unsupervised Learning

Drymonas et al. [26] from the Technical University of Crete designed **OntoGain** system for the unsupervised learning of ontologies from unstructured text in medical and computer science domains. OntoGain also uses linguistics and statistics-based techniques for acquisition of ontology. It uses The OpenNLP suite of tools and the WordNet Java Library for preprocessing of text such as tokenization, lemmatization, pos tagging, and parsing. To build a hierarchy agglomerative clustering and Formal Concept Analysis is implemented. For this initially each term is considered to be a cluster and these clusters are merged at each step based on the similarity measure. Then a formal context matrix is constructed which contains multiword terms and verbs. This matrix is given as input to Formal concept analysis algorithm. Association rule mining is used to extract the non-taxonomic relations.

v) Document Based Ontology

Jizheng Wan et al. [6] have proposed the concept of Document based Ontology (DbO) for constructing ontology from unstructured text which gives importance only to the properties of a document ignoring their context. The concept structure and entity instances are taken care of in this work. Statistical techniques such as Latent Semantic Analysis and Markov Model are used for detecting synonyms and to predict next word.

vi) Re-engineering and Reusing Resources

Mari Carmen et al. [9] suggests multiple ways of building ontology instead of giving any one methodology for different situations by re-engineering and reusing knowledge resources

vii) Glossary Based Approach

GOSPL [8] given by **Christophe Debruyne** has aimed at building hybrid ontology where concepts are defined both formally and informally. This methodology provides collaboration between the ontology engineers and domain experts and uses Glossary as a special linguistic resource. Structural natural language is involved as a vehicle to extract all the relevant and useful concepts from communication among community and these social processes are mapped to the processes of evolution of the emerging ontology. The concept of “sameness” is explored in detail according to which different terms from different communities referring to same concept do not imply to be synonyms.Fact modelling by applying the principle of separation in conceptualization is used which is an interpretation process called reasoning. The approach provides only the setting in which ontologies can be built but they have not given the method how community can use it.

viii) Probabilistic Modelling

Hoifung Poon and Pedro Domingos [27] have given an approach that overcomes the problem of inducing ontology from individual words by focusing on phrasal verbs etc. This approach is different from existing approaches as the ontology is induced probabilistic modelling to reduce uncertainty and noise. Also knowledge extraction and ontology population go hand in hand. Unsupervised Markov logic Network is used in this approach to form hierarchical clustering from logical expressions having is-a relations among them. The *is-a* relation among relation cluster can be found among relation clusters but it fails to do same for entity clusters. Active voice is well handled here but they are not able to handle passive voice. Moreover semantic relations are not extracted nor can this approach scale up to large corpora.

ix) Naȉve Bayes Classification

G. Suresh Kumar [28] has proposed an approach to extract concepts and relations for a question answering system in which domain attributes and associations are extracted from relevant documents. A binary decision tree-based rule engine is proposed giving output as a triplet of candidate keyword, predicate and associated object. The triplet extracts feature data which is given as training set to Naȉve Bayes Classifier. The relation between the concepts is through the relation predicted by classifier. Lexico-syntactic probability and lexico-semantic probability are used here. But only pre-classified classes of relations can be there.

Julia Hoxha et al. [29] also use Naȉve Bayes Classification for constructing ontology. The classifier is used for categorizing text to determine the label of document. SVM is used to cluster similar documents. In this method summarization is performed to shorten the text. Taxonomical relations such as synonyms, hypernyms and hyponyms are extracted here. Hearst's patterns are used for extracting hierarchies from text. At first candidate classes are extracted and then other hypernym, synonyms are extracted and represented in the form of taxonomy.

x) Machine Reading And Lexico Semantic Method

Bothma [13] gives a semi-automated approach for learning ontologies from Swedish text. Machine reading, statistical and lexico- semantic methods are used to extract concepts, a few taxonomic and a pre-defined set of non-taxonomic relations. This approach is also error prone as noun phrases are not taken into account to be extracted as concepts. Also no consideration is given to attributes of concepts.

xi) Rapid Prototyping

UPON-Lite [7] by **Missikoff et al.** is an automatic ontology development methodology that insists there should be no intermediation of ontology engineers in the process of building ontology. The methodology which is user centric is based on an incremental process that constructs rapid prototypes of trial ontologies. The role of domain experts is emphasized while the ontology engineers only formalize the ontology at the time of its release before the users. Researchers suggest the use of supporting tools like gloss extractors and ontology editing tool like Protégé for producing OWL ontology.

xii) Hierarchical Semi-Supervised Classification

Bhawna Dalvi [30] have proposed a hierarchical semi-supervised classification approach completes the incomplete class hierarchies by adding new instances to the existing ones or by discovering new classes and extending the existing ontology by placing them at appropriate places in the ontology using. This approach can be used for document classification task and entity classification into class hierarchy of a knowledge base also but is not applied to class-hierarchies that are non-tree structured.

xiii) Statistical, Machine-Learning, And Custom Pattern-Based Method

Open Calais [31] by Marius-Gabriel system by Thomson Reuter's which is linked to a market leading ontology extracting entities (persons, events, places), relationships etc and gives results in *rdf* format. The semantic content of users' input files is analyzed using a combination of statistical, machine-learning, and custom pattern-based methods. It also maps the metadata-tags to Thomson Reuters unique Ids supporting disambiguation and linking of data across all the documents being processed by it.

xiv) Topic Modelling Algorithm

Monika et al. [32] explores topic modelling algorithms such as LSI & SVD and Map Reduce LDA (Mr. LDA) for learning Ontology. The study and experimental result give enough proof of the effectiveness of using Mr. LDA topic modelling for learning ontology. Experimental results in the paper demonstrate the effectiveness of the proposed system in term of building richer topic-specific knowledge and semantic retrieval. Terminology ontology building is a preliminary step for semantic-based query (Topics and Words Detection) optimization for knowledge management. Their method is scalable but requires human intervention.

xv) User Centric Approach

A methodology is proposed by **Kenneth et al**. [33] that performs user-centric ontology population that needs human intervention at each step as the user is required to assist in developing, linking and maintaining the conceptualization of that domain, making the use of some already available ontology. Three main steps are followed where the first one is to select the relevant ontologies, then aligning the concepts with the same of the target ontology using a new hierarchical classification approach and after that user is assisted to develop, replace or enhance their initial ontology by creating, splitting or merging the concepts or adding new instances to existing concepts by extracting new facts from unstructured data.

LexOnt [34] by **K. Arabshian** is a system that also constructs the ontology semi-automatically including user at each step. It uses Wikipedia, WordNet and Programmable Web directory of services. It also uses existing ontology to extract relevant terms. LexOnt constructs the ontology in iterations, by interacting with the user. The user has the ability to choose, add terms to the ontology and rank those terms. It is a plug-in tab for the Protégé ontology editor. The system accepts unstructured text as input and *interacts with the user to facilitate the ontology creation process.*

xvi) Morpho-Syntactic analysis

Sourish Dasgupta and Jens Lehman [35] build initial ontology using the fundamental knowledge about the target domain. A corpus of text relating to that domain is analysed syntactically and semantically to perform semantic enrichment. Morpho-syntactic analysis of the text is done to extract concepts for building ontology.

xvii) Concepts Maps

OntoCmaps [36] by **A. Zouaq** is a domain independent ontology learning tool. It extracts deep semantic representations from corpora. It generates conceptual representations which are in the form of concept maps. This tool relies on the inner structure of graphs to extract the important elements that are identified as the important concepts. *This system is not able to capture non-hierarchical relationships.*

All these discussed works are summarized in Table 2.1 along with their methodology and problems.

Table 2.1: Methodology And Deficiencies In Existing Works

Technique proposed by	**Methodology Used**	**Source Text**	**Level of Automation**	**Deficiency**
Khurshid Ahmad et al.	Statistical techniques and Hearst's patterns	Unstructured Text	Semi-Automatic as Human intervention required as domain expert to provide evidence	Non-hierarchical relationships are not captured
Marius et al.	Open Calais	Unstructured Text	Automatic	A fixed set of Non-hierarchical relationships are captured
Hristo Tanev et al.	Weakly Supervised Approach	Unstructured Text	Semi-automatic	Non-hierarchical relationships are not captured
Jiang and Tan	Concept-Relation-Concept Tuple-Based Ontology Learning	Unstructured Text	Semi-automatic	Domain specific, may extract erroneous concepts or relations
Drymonas et al.	agglomerative hierarchical clustering and formal concept analysis, association rules and conditional	Unstructured Text	Semi-automatic	Ambiguous terms may get extracted, fixed set of non-hierarchical relations

	probabilities			
Jizheng Wan et al.	Document Based Ontology	Unstructured Text	Semi-automatic	Context of the document is not considered. Only properties of the document are taken care of.
Monika et al.	Topic Modelling	Unstructured Text	Semi-automatic	Non-hierarchical relationships are not captured
Bhavna Dalvi	Hierarchical Semi-Supervised Classification	Unstructured Text	Semi-automatic	Non-tree structured class hierarchies not explored
Christophe Debruyne	Glossary as linguistic resource Based Approach	Unstructured Text	Semi-automatic	No guidelines for communities to build the ontology, just a setting is provided.
Mari Carmen Sua´rez-Figueroa	Re-engineering and Reusing Resources	Unstructured Text	Semi-automatic	Methodologies and guidelines only are provided
Hoifung Poon and Pedro Domingos	Probabilistic Modelling (Markov Logic and Hierarchical Clustering)	Unstructured Text	Semi-automatic	Passive voice not handled properly. Semantic relation not extracted, non-scalable to large corpora
G. Suresh Kumar	Naı̈ve Bayes Classification	Unstructured Text	Semi-automatic	Non-hierarchical relationships are not captured
De Nicola & Missikoff	Rapid Prototyping	Unstructured Text	Semi-automatic as domain expert is needed at each step	Non-hierarchical relationships are not captured
Julia Hoxha et al.	Naı̈ve Bayes Classification along with Hearst’s Patterns	Unstructured Text	Semi-automatic	Fixed set of relationships are captured
Kenneth Clarkson	User Centric Approach	Unstructured Text	Semi-automatic	Non-hierarchical relationships are not captured
Sourish Dasgupta and Jens Lehman	Morpho- Syntactic analysis	Unstructured Text	Semi-automatic	Only is-a relations are extracted
K. Arabshian	User Centric Approach, plug-in for Protégé	Unstructured Text	Semi-automatic	Non-hierarchical relationships are not captured
A. Zouaq	Concepts Maps	Unstructured Text	Automatic	Non-hierarchical relationships are not captured
Bothma	Machine Reading and lexico-semantic method	Unstructured Text	Semi-automatic	Fixed set of non-hierarchical relationships are captured, lacks in extracting the compound concepts.

As evident through the Table 2.1, most contributions in ontology building are not able to fetch all the semantic relations present in the text and also compound concepts are not extracted. Also a few works are able to generate ontology from unstructured text automatically.

2.4 ONTOLOGY ENRICHMENT APPROACHES

PACTOLE (Property And Class Characterization from Text to OntoLogy Enrichment) is an approach proposed in [37]. A collection of astronomy texts are given as input for which ontology is constructed and a set of new concepts and instances are given as output to be inserted in the initial ontology. The process of enrichment process is based on Formal Concept Analysis (FCA).This work is not able to annotate object whose nature is unknown.

Another approach to enrich ontology is given by **Navigli et al.** [38] that proposes to use online glossaries to enrich an existing ontology. The core ontology property specifications are provided with natural language definitions for each class and are converted into web ontology language.

Castano et al. [39] has also proposed a methodology for enriching an existing ontology by matching new knowledge extracted from data with the existing ontology and annotating it..

Maria Teresa [40] gives an approach to make the ontology so expressive that the concepts and their intended roles can be understood so that the information present in the ontology may be reused easily. The technique used by the researchers is semi-automatic which uses WordNet.

2.5 TEXT SUMMARIZATION APPROACHES

Summarization helps user to find meaningful and relevant information from large text documents as summary of a document may help readers to go through the most important aspects of the document instead of having to read the full-length document. Headlines, table of contents, abstracts, reviews, highlights etc. give the summarized view of a large text.

This section reviews some significant achievements that have been obtained in the area of document summarization. Some approaches summarize by finding the salient information by finding pair wise similarity between all sentences or by clustering sentences using some similarity score.

Different researchers have proposed many techniques to generate summary such as: using features, using graphs as a collection of sentences as nodes & the edges denoting the similarity among sentences or by using cluster as a similarity measure or by using knowledge base. These approaches may be divided into several categories:

- Graph based approaches
- Ontology based approaches
- Machine learning based approaches

In the following subsection, the details techniques related to these type are presented.

2.5.1 Graph Based Approaches

Graph-based approaches for sentence-based summarization generate a graph in which the document sentences are represented by nodes and weight on each of the edge is calculated by a similarity measure that has to be evaluated on each node pair.

Leskove [41] generated document summary by using a semantic representation of the document and machine learning to create semantic sub-structure that can be used for extracting summaries. This approach shows the importance of the document semantic structure attributes in the sentence selection process. This can be used for abstract summary creation for a single document as well as multi document where linguistic features optimize the performance when training data contains shorter summaries while semantic features do the same for longer summaries.

In [42]Archetypal analysis and weighted archetypal analysis is used by **Canhasi**, to compute the positive and negative sentences for a given graph representation of a document set. Clustering and matrix factorization is also used in this approach.

Parveen et al. [43] features a method to extract single document summary by making bipartite graph consisting of sentence and entity nodes. Sentences are ranked using a graph based ranking algorithm. Redundancy is removed and the sentences are checked for their local coherence and summary is generated. Very little linguistic information is contained in the entity graph. In this method human subjects are included as judges to analyze the performance instead of domain experts that could give better judgment.

Another approach discussed in [44] by **Siddhartha Banerjee** takes the sentences from important documents and are aligned to sentences in other documents generating clusters of sentences that are similar. A word-graph structure is made from the sentences in each cluster and K-shortest paths are generated for this graph. Integer linear programming is used to select sentences having the shortest paths.

2.5.2 Ontology Based Approaches

Some researchers have made efforts to utilize ontology to make the process of summarization better. Documents related to same domain may share the same information and can use the same domain ontology for this purpose. Ontology helps to extract the entities and categorization of sentences.

Ragunath [45] presents an idea for ontology-based summarization to compute a set of features for each sentence based on the output of the hierarchical classifier. This is an extractive summarization approach in which existing sentences or phrases are selected from original text. In this approach a sentence is classified to a leaf node or to an internal node. Nodes sharing common sub trees are matched using the classifier. If a sentence is mapping to more than one sub tree in the hierarchy, all nodes from each sub tree is included. For each sentence confidence weights assigned by the classifier are used to compute a sub tree overlap measure. Only hierarchical ontology is used here for sentence mapping.

Hennig [46] also use a hierarchical ontology to generate summaries. Their work maps the sentences of original document to the nodes of the ontology using an SVM classifier which is trained using search engines for sentence classification. The mapping of nodes to the ontology gives a semantic representation of content of document sentence which improves the quality of summaries. It computes structural properties of the hierarchy and category labels using sentence features to improve summarization. Only a small ontology instead of some bigger non-hierarchical ontology is utilized here.

Ontological knowledge is used by **Verma et al.** [47] also to generate document summary. Query based summary is generated which utilizes WordNet or UMLS ontological knowledge to revise the query and then by calculating the distance of query from each sentence. The sentences having lesser distance than a threshold are included in set of candidate sentences to be included in summary. These sentences are again divided into groups by calculating the pair wise distances among them and then the highest ranked sentences are chosen for the final summary. Natural language programming techniques are not used here so in the absence of syntax analysis, grammatically incorrect sentences from original documents hamper the quality of summaries. It is an extractive approach as abstract statistical data is also not utilized for summarization. In this method redundancy reduction in the summary is also not up to mark as it covers same information from multiple documents.

Baralis [48] depends upon YAGO ontology to evaluate and select sentences from documents. Entity recognition and disambiguation steps in the process of generation of document summary are performed using YAGO which identifies the key concepts in the document and their significance is evaluated with respect to the context of the document. The problem with this method is that the summaries generated by this method are less compact.

2.5.3 Machine Learning Based Approaches

These approaches learn a model that determines importance of sentences using a training corpus of full texts and their summaries. Different models such as Naĩve Bayes, Decision trees, SVM, HMM, CRF can be used. These approaches operate at lexical level and provide good results for query based summaries. The features taken into account by these models can be Sentence length, presence of indicator phrases, Sentence position (first/medium/final), highly weighted content words, Containment of (important) named entities, Containment of specific topic words. A large amount of text is needed for earning purpose as human-generated summaries are required to train a classifier for the given text. These approaches are unable to manipulate information at abstract level.

A.P.S et al. [49] give an approach for single document summarization that uses two measures to evaluate importance of a sentence: first is the frequency of the terms in the sentence and the other is the similarity to the other sentences. The sentences in the document are ranked according to their respective scores and the top ranked sentences are selected for summary. The statistical sentence selection measures include: Sentence position, Cue words, Document frequency, Inverse document frequency, Term frequency. Their approach uses nearest neighbor search technique to find the neighbor documents that are similar to the specified document. The sentences are scored using global affinity graph. The highest scored sentences are then checked for redundancy at the document level.

Patil et al. [50] shows that the choice of the classifier influences the performance of the trainable summarizer strongly. The procedure of automatic trainable summarization employs statistical and linguistic features which are extracted directly and automatically from the original text.

Babara [51] uses latent semantic analysis and fuzzy logic system to extract the summaries from the original text. A set of features is used which includes the title sentence, sentence length, sentence position, numerical data, proper nouns etc. Each feature is given a score using fuzzy logic. Based on this score each sentence is classified into three classes of important, average and unimportant and are thus selected to create the summary. This approach is not applicable for multi document summary.

Kaliappan [52] uses the Naïve Bayesian Classification and the timestamp concept. This summarizer may work on many domains as it does not uses knowledge base. The user can specify the compression rate so that amount of information to be extracted from the documents can be chosen.

Singh et al. [53] have presented a technique using unsupervised deep learning approach to summarize documents from Hindi and English. A set of eleven features is extracted from each sentence of document to generate the feature matrix which is passed through Restricted Boltzmann Machine to increase accuracy of choosing relevant sentences.

All these discussed works are summarized in Table 2.2 along with their methodology and problems.

Table 2.2: Comparative Study of Text Summarization Approaches

Approach	Methodology	Technique Used	Type of Summary
[42]	Archetypal analysis and weighted archetypal analysis	Graph Based	Multi document extractive summary
[41]	Semantic substructure	Graph based/machine learning	Abstract summary/multi document summary
[43]	Bi-partite, ranking algorithms	Graph based	Single document extractive

			summary
[44]	Clustering of	Graph based	Multi document extractive summary
[51]	Latent semantic analysis/fuzzy logic	Machine learning	Multi document extractive summary
[48]	Yago ontology	Ontology based	Multi document extractive summary
[46]	Hierarchical ontology	Ontology based	Multi document extractive summary
[47]	WordNet and UMLS	Ontology based	Multi document extractive summary
[49]	Nearest neighbour search	Machine Learning	Single document abstractive summary
[45]	Sentence features	Ontology based	Single document abstractive summary
[50]	Statistical and linguistic features	Machine Learning	multi document extractive summary
[52]	Naïve Bayesian Classification and the timestamp concept	Machine Learning	multi document extractive summary
[53]	unsupervised deep learning	Machine Learning	Bi-lingual, multi extractive document summary

As it can be seen from the Table 2.2, most of the contributions focus on extractive summarizations that are able to make the text concise but may contain redundant information. The techniques generating abstractive summaries that do not exploit the semantic structure of the sentence are prone to generate erroneous results.

2.6 TECHNIQUES, TOOLS AND KNOWLEDGE STRUCTURES

This section presents the introduction of key tools and platforms to support the designing and the development of the ontological framework and the other proposed techniques for ontology enrichment and text document summarization. Specifically, natural language processing tools such as dependency parser (Stanford Parser), name entity recognizer (SENNA), anaphora resolver (Java RAP) [54] and some other tools such as GraphViz, SVM Classifier are introduced. WordNet is presented here as the knowledge structure.

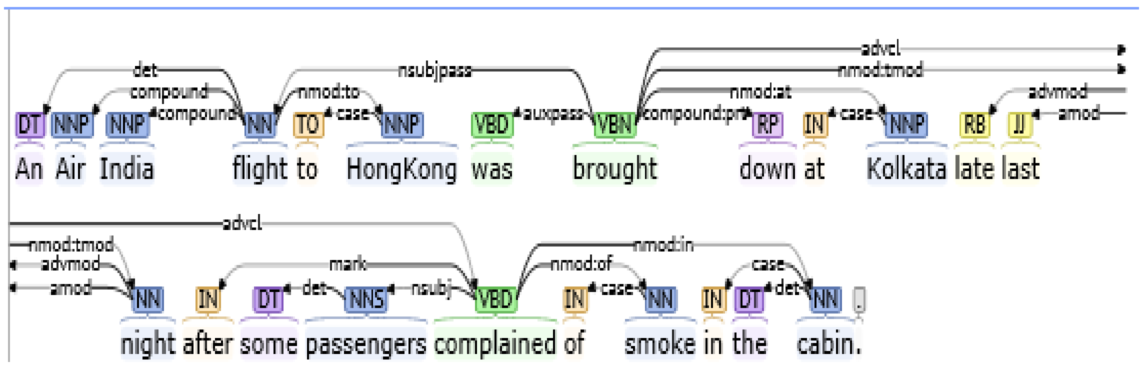
2.6.1 Dependency Parser

In natural language parsing techniques, dependency based representations has found their potential use in disambiguation. The lexical elements form a syntactic structure having interlinks of binary asymmetrical relations between the words of sentence. These relations are called dependencies. In natural language processing systems, dependency structures are quite expressive to be convenient and restricted enough for allowing full parsing providing sufficiently high accuracy and efficiency.

We have used Stanford Dependency Parser [55] in our work. A brief description of this parser is given in as follows:

Stanford Dependency Parser

To extract the textual relationships without the help of linguistic expertise, the Stanford typed dependencies representation were designed so as to provide a simple description of the grammatical relationships in a sentence so that people can easily understand and effectively utilize these relations. All the words in a sentence are connected to each other with some grammatical relations like ‘subject’, ‘modifier’, ‘determiner’, and so on. These relations are known as dependency relations as they express how one word is dependent on another word. In a dependency relation between two words, one is called DEPENDENT, which generally acts as modifier, object or complement of the other word, known as HEAD. Figure 2.2 gives an example of a dependency tree (which is basically a directed acyclic graph with arcs pointing from the HEAD to the DEPENDENT). The arc-labels (also known as attachments) represent the dependency relations. The idea here is to parse the sentences of a document using Stanford dependency parser The parser provides dependency tags attached with each term which are used to generate new tags Roles, Properties and Hierarchy.



DT NNP NNP NN TO NNP VBD VBN RP IN NNP RB JJ
An Air India flight to HongKong was brought down at Kolkata late last
det compound compound nmod:to case nsubjpass auxpass advcl nmod:tmod nmod:at compound:prt case advmod amod
NN IN DT NNS VBD IN NN IN DT NN .
night after some passengers complained of smoke in the cabin.
advcl nmod:tmod advmod amod mark det nsubj nmod:of case nmod:in case det

Figure 2.2: Dependency Tagged Sentence

2.6.2 Named Entity Recogniser

Named Entity Recognition is one of the very helpful information extraction techniques to recognize and classify named entities in text. These entities are categorised to some pre-defined categories such names of persons, locations, organizations, time representations etc. There could be other specific terms also apart from these generic entities, which could be defined for a particular problem. These terms represent segments or elements having a unique context in the text. To categorize such custom entities, machine learning models could be trained. These entities are generally denoted by proper names so mostly noun phrases in text documents indicate them.

NER Categories

NER has three top-level categories:

i) Entity names:- Entity Names represent the element’s identity, for example name of a person, title, anything living or non-living etc.

ii) Temporal expressions:- A temporal expression is where time related events are shown by some sequence of words for example times of day, durations, calendar dates etc.

iii) Number expressions:- A mathematical sentence involving just numbers and/or operation symbols denotes number expression.

Approaches to NER

There are two different approaches to implement NER chunker to tag specific elements in the text. One approach is knowledge/rule based and the other approach uses supervised machine learning. A combination of both approaches may give better results for some problems.

- **Knowledge/ rule based approaches**

A rule based NER system uses some predefined language dependent rules based on linguistics for identification of named entities in a document. These systems perform well but have limitations that these are not flexible to changes. The entities found by these systems generally are proper nouns or proper nouns in alliance with numbers.

- **Machine Learning approach**

Machines can predict custom entities on a given text using supervised machine learning on labelled data.

There are many named entity tagger available. We have utilized SENNA as name entity tagger in our framework as SENNA achieves close accuracy with Stanford pipeline twice the speed and less memory usage.

To perform a number of NLP tasks accurately and speedily, SENNA (Semantic/syntactic Extraction using a Neural Network Architecture) [56] is employed which is a tool having multilayer neural network architecture. These tasks include part-of-speech tagging, chunking, named entity recognition, and semantic role labelling. SENNA extracts essential features from unlabelled text using deep learning. Auto-encoders and neural networks language models are used to perform the unsupervised learning phase. The pipeline maps words into other space of representation having lower dimensionality. SENNA has dictionary of 130 thousand words which is used to map every word to a vector of 50 floating numbers. Convolutional networks are used to merge these vectors into a sentence structure. Different classifiers are generated by training the same architecture for different tasks using annotated text. The major advantage of this approach is that lesser amount of engineering is required to solve multiple problems. A lot of prior knowledge is not used by SENNA. The features used by the system were only pre-trained word embeddings, gazetteer list and uppercase information.

IOBES tagging scheme

To mark a noun phrase containing a single word, the tag "S-NP" is used.

"B-NP", "I-NP", and "E-NP" tags are used to mark the first, intermediate and last words of the noun phrase.

"O" is an additional tag that marks words which are not members of a chunk.

Four different types are defined:

- Person(PER),
- Organization(ORG),
- Location(LOC)
- Miscellaneous(MISC)

Example of NER tagging by SENNA is as follows in Figure 2.3:

Columbia/ORG is an American/Misc university located in New/LOC York/LOC.

Figure 2.3: Annotated Text After NER

Here in this example ORG, refers to the organization and LOC denotes the Location.

2.6.3 Anaphora Resolution

The Process of finding the antecedent for an Anaphor is Anaphora resolution. Here anaphor is the reference that points to some previous item and antecedent is the entity to which that anaphor refers. For example

Mona Singh says she will always be grateful to Anu Malik. The actress revealed that the musician helped her calm down when she became scared by a thunderstorm while travelling by a plane.

In the above given text anaphors and their antecedents will be as shown in Figure 2.4:

Anaphor		**Antecedent**
She	=>	*Mona Singh*
The actress	=>	*Mona Singh*
The musician	=>	*Anu Malik*
Her	=>	*Mona Singh*
She	=>	*Mona Singh*

Figure 2.4: Resolved Anaphora and Antecedent

Anaphoric reference resolution is quite challenging task in Natural Language Processing field. It is very difficult to give a complete, reasonable and calculable description of the resolution process, because of the unawareness of the particularities. Anaphora resolution needs to be addressed in most of the applications dealing with natural language e.g. information extraction, machine translation systems or dialogue systems. There are following approaches to anaphora resolution namely

- Rule Based
- Machine Learning Based
- Statistical Based

The context of the expressions define the interpretation of these expressions

2.6.4 Jena API

For creating and manipulating RDF graphs object classes called interfaces are provided by a Java ontology API known as Jena. The graph generated by Jena is called a Model having extension *rdf* and is represented with the Model interface. RDF statements are described by the resources, properties and literals given as the Resource, Property and Literal interfaces. Several methods are provided by Jena that allow RDF graphs to be saved and retrieved to and from files. Various database management systems are supported by Jena such as MySQL, PostgreSQL, Oracle etc. Also various tools including such as a parser and I/O modules for RDF/XML output are provided by Jena.

2.6.5 GraphViz

To generate a graphical presentation of the ontology generated by our system we have made use of graph visualization tool named as GraphViz [57]. In this section we will describe briefly the function of this tool. Graph visualization is used for representing structural information of abstract graphs and networks. Automatic graph drawing has many significant applications in various fields such as software engineering, web design, database, networking and in many other visual interfaces for different domains.

GraphViz, which is an open source graph visualization software contains several main graph layout programs. It also contains other web and interactive graphical interfaces, some auxiliary tools and different libraries. GraphViz makes diagrams by taking descriptions of graphs in a simple text language.

The diagrams can be made in many formats such as images (png, jpg) or SVG for web pages, GraphViz has many useful features and options for colors, many fonts, tabular node layouts, different line styles, hyperlinks, and many other custom shapes. Graphs may be created and edited manually or from external data sources either as raw text files or within a graphical editor.

GraphViz follows a general hierarchical approach for drawing graphs. It works in four phases. The first phase breaks any cycle that occurs in the input graph. The second phase assigns nodes to discrete levels or ranks. In the next phase ordering of nodes is done within ranks so as to avoid crossings. The fourth phase sets X coordinates of nodes so that edges are short. GraphViz accepts input in the DOT language. Three major kinds of objects namely graphs, nodes and edges are described by this language. The main graph is a directed graph (digraph). The following program as shown in Figure 2.5 written in DOT language defines a directed graph. Nodes' shapes, labels, colors, styles are defined, Ranks of nodes, separation distance and their direction is also specified in this program.

```
digraph mygraph{
  passengers ->accommodated
  passengers [shape=box,style=filled,color=Green];
  passengers -> agent_S3 [style=bold,label="HAS ROLE"];
  agent_S3 [shape=box,style=filled,color=Pink];
  accommodated ->hotels
{ rank= same rank sep=1.2  rankdir=LR; passengers hotels  }
  hotels [shape=box,style=filled,color=Green];
  hotels -> theme_S3  [style=bold , label="HAS ROLE"];
 theme_S3 [shape=box , style=filled , color=Pink];
}
```

Figure 2.5: dot file for GraphViz

The graph corresponding to this dot file will be generated by GraphViz using the command line or with a graphic visualization service that may be web based or other GUI based interface as shown in Figure 2.6.

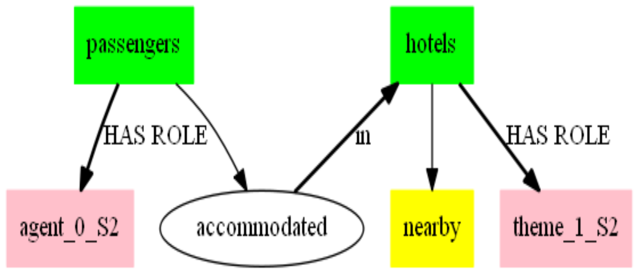


Figure 2.6: Graph Generated for dot File

2.6.6 SVM Classifier

Support Vector Machines given by [58] are among the best supervised learning algorithms which provide a powerful approach even in case of high dimensional feature space. Joachims T. [59] used SVM classifiers for text categorization.

Considering its linear form for a binary problem with feature x and label $y \in \{-1, 1\}$

Training data is represented as $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)$. We define the maximum margin hyperplane as

$$\vec{w}.\vec{x} + b = 0 \qquad \text{Equation 2.1}$$

Where $\vec{w}$ is the normal vector to the hyperplane.

Two parallel hyperplanes are determined that separate the two classes of data, so that the distance between them is maximum. The region bounded by these two hyperplanes is called the "margin", and the maximum-margin hyperplane lies halfway between them as shown in Figure 2.7 .

$$\vec{w}.\vec{x} + b = 1 \qquad \text{Equation 2.2}$$

$$\vec{w}.\vec{x} + b = -1 \qquad \text{Equation 2.3}$$

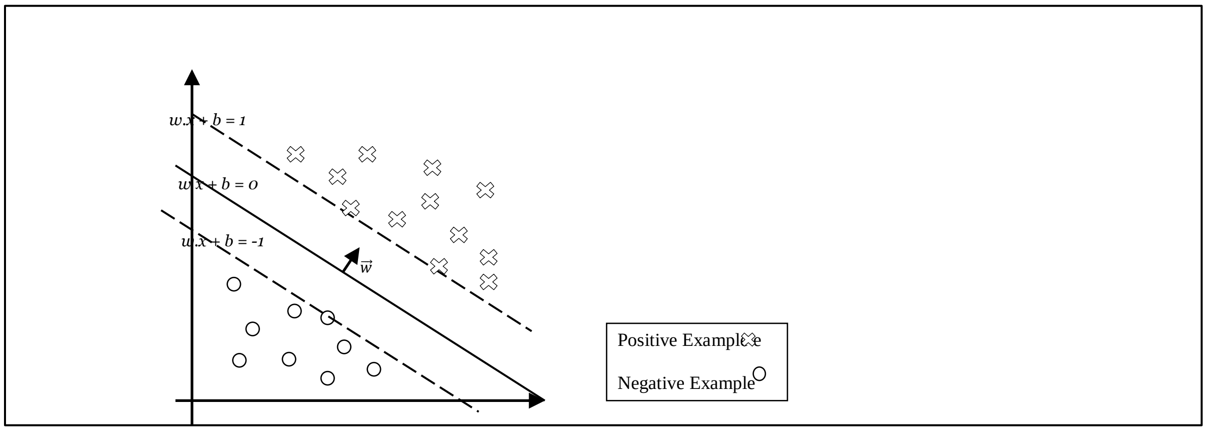


Figure 2.7: SVM Classifier

To make the algorithm work for non-linearly separable datasets, the optimization is explicated as:

$$min_{w,b}\ \frac{1}{2}\ ||w||^2 + C\sum_{i=1}^{m}\xi_i$$

Such that

$\boldsymbol{y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1 - \ \square_i\ ,\ \ i = 1, \dots, m}$ Equation 2.4

$\xi_i \geq 0, i = 1, \dots, m$

where *ξi* is called slack variable.

The $\frac{1}{2}\ ||w||^2$ specifies size of the margin and second $C\sum_{i=1}^{m}\xi_i$ specifies misclassification.

Training an SVM involves the reduction of above equation to a NP problem from which decision function can be derived.

$\boldsymbol{g(x) = \sum_{i=1}^{l}\lambda_i y_i x_i\ . x + b}$ Equation 2.5

where the parameter determines the trade-off between increasing the margin-size and ensuring that the lie on the correct side of the margin.

Sentence Extraction using polynomial kernel

For a non-linear decision surface, kernel trick is applied to maximum-margin hyperplane and the dot product is replaced by the kernel function.

$$\boldsymbol{k}(\vec{\boldsymbol{x}}_{\boldsymbol{i}}\,, \boldsymbol{y}) = (\vec{\boldsymbol{x}}_{\boldsymbol{i}}\boldsymbol{y} + \boldsymbol{1})^{\boldsymbol{2}} \qquad \text{Equation 2.6}$$

This polynomial kernel has been very effective when applied to several tasks of natural language processing of a second degree with a value of C as 0.0001. We have used the same for extracting summary from text. This extractive summary is further shortened by constructing ontology of this extractive summary using the proposed system for generating ontology and reconstructing the sentences using this ontology to generate abstractive summary.

2.6.7 WordNet

WordNet [60]is a general-purpose online lexical semantic electronic repository for the English language. Its structure, characteristics and potential usefulness is described here in this section.

WordNet provides a thesaurus and lexicon, semantic bond among the majority of English terms. It classifies words into categories and inter-relates the meanings of those words. It organisation is in the form of synonym sets (*synsets*) which are set of words that are can be interchanged according to some context, as they share a commonly-agreed upon meaning having little or no variation. There may be different senses of each word in English in which the word may be interpreted and each of these distinct senses denotes different *synsets*. There is a pointer to at least one *synset* for every word.

The *synsets* can be thought of as nodes in a graph where a semantic pointer is a directed edge in the graph. The pointer has one end source and the other end a destination.

Some semantic pointers which are useful are:

- hyponym: if X is a (kind of) Y, X is a hyponym of Y
- hypernym: if Y is a (kind of) X, X is a hypernym of Y
- part meronym: if X is a part of Y, X is a part meronym of Y
- member meronym: if X is a member of Y, X is a member meronym of Y
- similar to: if the two synsets have meanings that are quite similar to each other, a synset is similar to the other one.

Each *synset* also contains a description of its meaning which is expressed in natural language known as gloss. WordNet also contains example sentences of usage of that *synset*. The information provided gives summary of the meaning of a particular concept and gives knowledge for a particular domain.

In our work, WordNet is used to link terms with its meaning (semantic annotation) in order to be able, for example, to extract similar terms for a given term exploiting its hyponyms, hypernyms, and synsets.

2.7 CONCLUSION

The analysis of the reviewed approaches allows us to conclude that these methodologies for building ontology from unstructured text are able to capture hierarchical relationships or some fixed set of non-hierarchical relationships among the concepts of the text but they fail to extract all the semantic relations present in the text.

We reviewed some approached to enrich the ontology so that the ontology can be extended with specific knowledge to provide more information about the constructs of ontology.

We present the review and compare various techniques for summarization of documents to get a concise form of the information in this domain. This review opens up new challenges to be taken ahead in the field of document summarization such as text coherency must be ensured as sentences may have dangling co-references. Also summarizing non-textual data, handling text from multiple sources effectively and getting good reduction rates are needed. The most difficult challenge is to achieve human quality summarization.

CHAPTER 3
AUTOMATIC ONTOLOGY CONSTRUCTION USING CONCEPTUALIZATION AND SEMANTIC ROLES

3.1 INTRODUCTION

Semantic web is a major evolution in connecting information for effective information retrieval. The goal of semantic web is to make the web understandable by both human and machine. This task is done by using ontologies as it is the better way to represent knowledge [36]. In other words, constructing ontologies aim at capturing domain knowledge that gives a commonly agreed understanding of a domain, which may be reused, shared among applications and groups.

In this chapter, we propose a new approach to build ontology automatically, based on extracting semantic roles present in the given sentences of a given text along with usual concepts and their relationships. The extracted information about different roles, concepts and relationships among the concepts from different sentences in the document are then merged to construct ontology for whole document. The proposed approach is implemented and the performance of the proposed technique is evaluated. Experiments show the ontology thus created captures most of the information given in the document. The present proposal may be important to understand the document as we have both syntactic and semantic information about a sentence or a text.

In general, process of building ontology adopts following steps. Firstly the concepts are extracted then underlying semantic relations (hierarchical or non-hierarchical) among these concepts are extracted and then these relations and concepts are connected using suitable criteria. However, the work done till now in automated creation of ontologies from plain text mostly capture only hierarchal relationships such as *car-(is-a)-vehicle* or *steering-(part-of)-car* but non-hierarchical relations such as performed, begin etc as in *Ghulam_Ali-***(***performed_at***)***-concert* or *music_festival-(begins)-tomorrow* are not captured accurately by the existing approaches [35] [14]. But the present approach uses semantic roles along with other components such as concepts and their relationships. The ontologies corresponding to each sentence in the given text is constructed and these ontologies are then merged by aligning the concepts in them to create a bigger ontology for complete document. More precisely, the semantic similarity technique is used to match and merge these structures related to semantic roles in addition to matching of relations and concepts. In particular, a set of ontology merging rules are designed and later used to merge the structures in the two different ontologies. The multiword concepts are also taken into account while identifying concepts and semantic roles. This way the limitations of the previous contributions are removed as those works were focusing mostly on single word concepts and taxonomical i.e. hierarchical relations.

3.2 BASIC APPROACH

The proposed technique takes unstructured text as input, applies natural language processing techniques to identify the concepts, roles etc. and utilises a new algorithms to merge them into an ontology. Therefore following textual components are playing key roles to design the present ontological framework.

i) **Semantic roles-** Semantic roles information along with other information such as concept and relations to generate ontology. Semantic roles are representations that express the abstract role that arguments of a predicate (usually expressed by verb in the sentence) can take in the event [61] [62]. For example a concept can be an *agent* or *accompanier* or a *location* in a sentence. Attaching these semantic roles with each constituent not only help to merge sub-ontologies (ontologies created for each sentence in the given document) but also contribute to deeper text understanding in the form of final ontology.

ii) **Concepts-** All concepts instead of just key concepts (frequently used concepts) are identified and used.

iii) Non-Taxonomical Relationships- Non-Taxonomical Relationships among concepts instead of just taxonomical relationships are used.

All these constituents will be identified and used in the sequence as is given in the architecture of the system in the following section.

3.3 ARCHITECTURE OF PROPOSED SYSTEM

In order to realize the basic approach or model computationally, following is the proposed architecture of the system as shown in Figure 3.1.

1. Natural language Processor

In this system the input text documents are first processed by a natural language processor which uses Stanford dependency parser [55] that performs the tokenization at sentence level and dependency parsing. This component also performs name entity recognition tagging and anaphora resolution and transforms.

2. Information Extractor

These sentences are transformed into tagged structures that are used by the Information Extractor module. This module extracts the required information i.e. concepts, relations, properties and semantic roles of concepts.

3. Sub-ontology Constructor

The information i.e. concepts, relations, properties and semantic roles of concepts extracted in above module is used in constructing intermediate structures or sub-ontologies for each sentence.

4. Sub-ontology Merger

These sub-ontologies are further required to be merged to form the complete ontology of the document. In our proposed ontology mapping and merging scheme, process of merging takes into consideration not only the shared concepts and similar relations but also the semantic roles each concept is playing in a sentence. A set of rules is used which is designed specifically for this purpose.

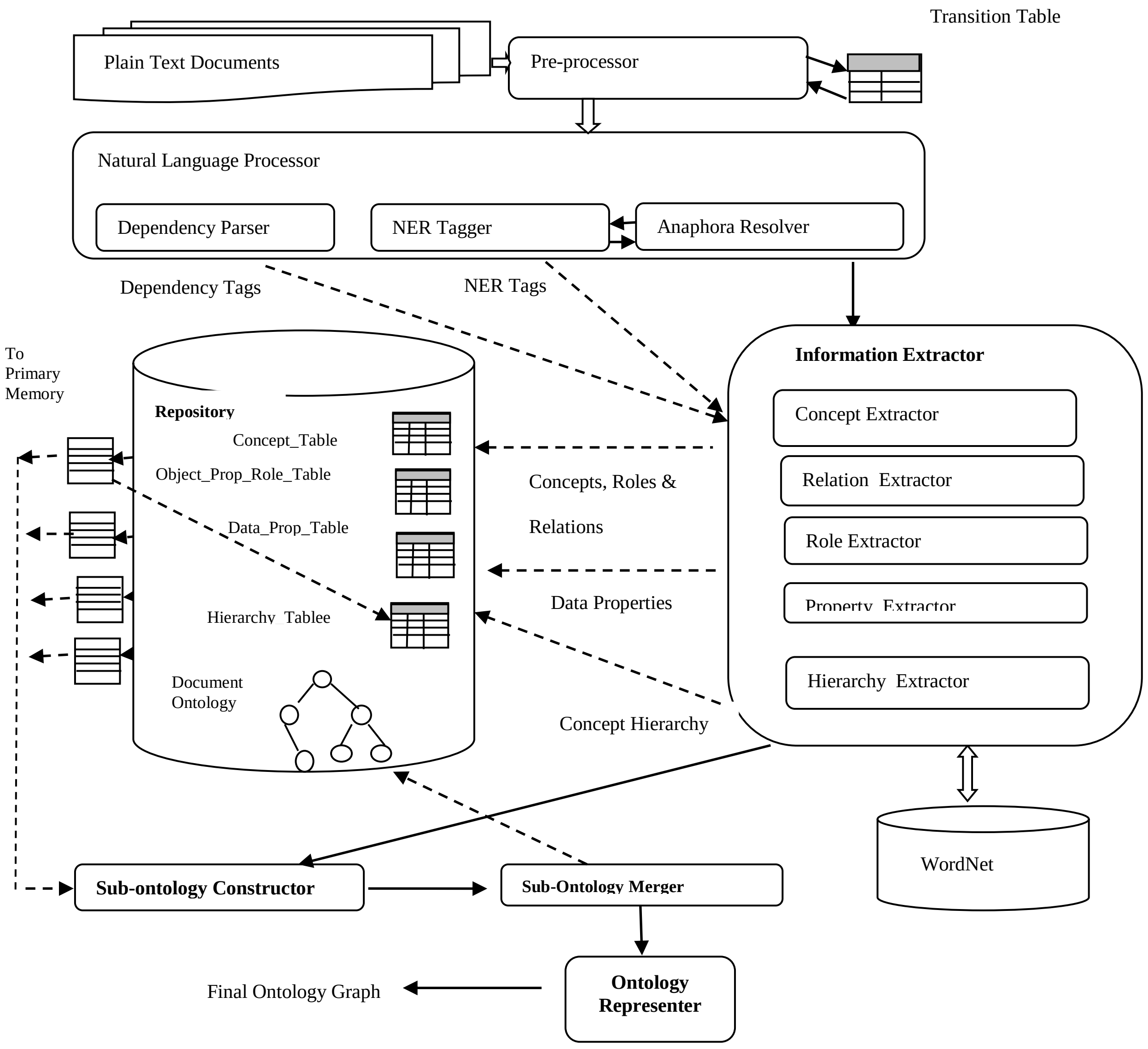


Figure 3.1: Architecture of the Proposed System

5. Ontology Representer

This module takes the bigger ontology, generated as result of above step, as input and represents it both graphically and as an *rdf* document.

The following section 3.4 focuses on the detailed design of the proposed system and also includes the data design and algorithm design of each module.

3.4 DETAILED DESIGN OF THE SYSTEM

In this section, detailed design of the components of the proposed system is given by using the following subsection. It may be noted that in the process of designing a component following two issues are taken in to account.

i. **Input data or Information required and its representation-** In the process of designing a component, first the data or information required is identified and then that data or information is represented using a suitable scheme by providing the format or structure of the data, storage of data and its utilization wherever required.
ii. **The algorithm-** For each component their respective algorithms are given to express the way of utilizing structure of the data or information to achieve a particular intermediate (or final) result.

In the coming subsections, we will give detailed design of each component of the system considering above two issues.

3.4.1 Pre-processor

The text documents being processed for constructing ontology may contain words, phrases or sentences which are redundant and unnecessary e.g. the phrases like "*as a matter of fact*", "*in all honesty", "considering"* and many more do not contribute towards meaningful information of the text and may be processed to be removed from this document. The sentences being followed phrases like "namely", "specifically", “thus”, “to put it another way” etc. just contain the redundant information and may be removed. To perform this task, different kind of transition words or phrases are first identified and then treated according to their type. The pre-processor takes the unstructured text as input and searches in the text for the transition words or phrases already stored in the dictionary. If a match is found in the text, it removes the transition word, or the transition phrase or the text following that word or phrase is removed. Action to be taken is decided according to the type of transition word or phrase e.g. for words like *such as, for instance* etc. which are Introductory type transition phrases, these words along with the text following these words is un-necessary and can be removed from the text without harming or distorting the information to be conveyed by the text.

- **Data Design for Pre-processor**

Since the present work is dealing with text written in English language, therefore, many types of transition words or phrases, found in the documents written using English language, are identified to take the appropriate action as stated below. These transition words are given in Table 3.1. The table also describes the type of these phrases or words and defines the action which can be taken if such types of transition words or phrases are identified by the pre-processor module.

Data Storage: This table is stored in secondary memory and is brought to primary memory as the first step of removing these un-necessary words.

Table 3.1: Transition Words Action Table

Transition word/ phrase type	**Example words**	**Action Taken**
Addition	indeed, further, as well (as this), either (neither), not only (this) but also (that) as well, also, moreover, what is more, as a matter of fact, in all honesty, and, furthermore, in addition (to this), besides (this), to tell the truth, or, in fact, actually, to say nothing of, etc.	Transition word/phrase removed

Introduction	such as, as, particularly, including, as an illustration, for example, like, in particular, for one thing, to illustrate, for instance, especially, notably, by way of example, etc.	Transition word/phrase removed. Also the text following the transition word/phrase in the sentence is removed.
Reference	speaking about (this), considering (this), regarding (this), with regards to (this), as for (this), concerning (this), the fact that, on the subject of (this),	Transition word/phrase removed
Similarity	similarly, in the same way, by the same token, in a like manner, equally, likewise, etc	Transition word/phrase removed
Clarification	that is (to say), namely, specifically, thus, (to) put (it) another way, in other words,	Transition word/phrase removed. Also the text following the transition word/phrase in the sentence is removed.
Conflict	but, by way of contrast, while, on the other hand, however, (and) yet, whereas, though (final position), in contrast, when in fact, conversely, etc.	Transition word/phrase removed
Emphasis	even more, above all, indeed, more importantly, Besides, etc.	Transition word/phrase removed
Result	as a result (of this), consequently, hence, for this reason, thus, because (of this), in consequence, so that, accordingly, as a consequence, so much (so) that, so, therefore, etc.	Transition word/phrase removed
Purpose	for the purpose of, in the hope that, for fear that, so that, with this intention, to the end that, in order to, Lest, with this in mind, in order that, so as to, so, etc.	Transition word/phrase removed
Consequence	under those circumstances, then, in that case, if not, that being the case, if so, otherwise	Transition word/phrase removed

Sequential Transition	in the (first, second, etc.) place, initially, to start with, first of all, thirdly, (&c.), to begin with, at first, for a start, secondly, etc.	Transition word/phrase removed
Continuation	subsequently, previously, eventually, next, before (this), afterwards, after (this), then, etc.	Transition word/phrase removed
Conclusion	to conclude (with), as a final point, eventually, at last, last but not least, in the end, finally, lastly, etc.	Transition word/phrase removed
Degression	to change the topic, incidentally, by the way, etc.	Transition word/phrase removed
Resumption	to get back to the point, to resume, anyhow, anyway, at any rate, to return to the subject, etc.	Transition word/phrase removed
Concession	but even so, nevertheless, even though, on the other hand, admittedly, however, nonetheless, despite (this), notwithstanding (this), Albeit (and) still, although, in spite of (this), regardless (of this), (and) yet, though, granted (this), be that as it may, etc.	Transition word/phrase removed
Summation	as was previously stated, so, consequently, in summary, all in all, to make a long story short, thus, as I have said, to sum up, overall, as has been mentioned, then, to summarize, to be brief, briefly, given these points, in all, on the whole, therefore, as has been noted, hence, in conclusion, in a word, to put it briefly, in sum, altogether, in short, etc.	Transition word/phrase removed

After deleting these transition words, the document contains lesser complex sentence structures and redundant information .

Data Utilization: These simple sentences are given as input to natural language processor so that the necessary information can be extracted as illustrated in the next section.

- **Algorithm Design for Pre-processor**

The algorithm for pre-processing of text documents is shown in Figure 3.2.

```
Algorithm pre_processor()
Input:  text document, dictionary of transition words ,Table 3.1
Output: pre-processed text document
Begin
      for each sentence of the text document
  1.  if any of the phrase from dictionary of transition words  is present in the
      sentence
  2.  find the type of the phrase
  3.  take action according to the Table 3.1
      end for
End
```

Figure 3.2: Algorithm for Pre-processor

The algorithm takes the text document as the input and begins by processing each sentence of the text document to check whether any of the phrases kept in Table have any occurrence in the sentence in step 1. If yes then step 2 finds the type of the phrase by looking into the table and takes the appropriate action like deleting the phrase or deleting the consequent sentence in step 3. The output of this algorithm is the pre-processed text document that is free from the un-necessary or redundant words or phrases.

3.4.2 Natural Language Processor

This section describes the processing of natural language processor which analyses input sentences from plain text documents syntactically and annotates the document with linguistic features that are needed by Information Extractor module. This module is having following components:

i) Anaphora Resolution

ii) Dependency Parser

iii) NER Tagging

These components are described as follows:

xviii) Anaphora Resolver

For extracting the correct information from text it is necessary to replace the pronouns in a sentence to its mention as a noun in some previous sentence. This process called anaphora or co-reference resolution is performed here using Java RAP tool (Qiu, Kan & Chua, 2004) which takes plain text as input and gives output in the form of plain text with in-place substitution of anaphora with its antecedent.

- **Data Design for anaphora Resolver**

We get the anaphora resolved sentences as output of this tool. The details for data design for anaphora resolver is as follows:

Output Data Format: As an example for the following sentence "*An Air India flight to HongKong was brought down at Kolkata late last night after some passengers complained of smoke in the cabin. The flight with passengers landed at the Kolkata airport. Under those circumstances, they were accommodated in nearby hotels."*

'*they*' will be replaced by "*passengers" as*

"*An Air India flight to HongKong was brought down at Kolkata late last night after some passengers complained of smoke in the cabin. The flight with passengers landed at the Kolkata airport. Under those circumstances, passengers were accommodated in nearby hotels."*

Data Storage: The anaphora resolved sentences are stored in *String* in primary memory.

Data Utilization: These anaphora resolved sentences are used by dependency parser module.

- **Algorithm Design for Anaphora Resolver**

The algorithm for anaphora resolver is shown in

Figure **3.3** as follows:

```
Algorithm anaphora_resolver()
Input: pre processed text document
Output: anaphora resolved text document
Begin
        for each sentence of the text document
        call Java RAP
        end for
End
```

Figure 3.3: Algorithm for Anaphora Resolver

As shown in algorithm anaphora_resolver, we pass the preprocessed text document as input to the Java RAP tool. The tool processes it and gives the anaphora resolved text document as output.

xix) Dependency Parser

Documents as plain text are given to Stanford dependency parser [dep] [55] which provides the result in the form of a part of speech tagged sentences of each document along with the dependencies among the constituents of each sentence. In this dependency graph vertices are the words in a sentence and an edge exists between each word and its syntactic head. The graph forms a tree rooted at the main verb. The edges are labelled with dependency types. These dependencies are utilized to find the concepts, relations and properties of concepts from the text.

The Stanford dependency tagging for the first sentence will be given as in the Figure 3.4.

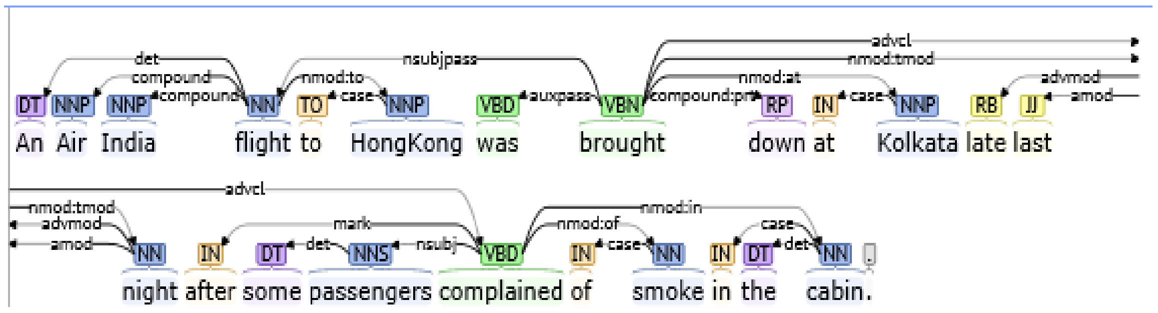

An Air India flight to HongKong was brought down at Kolkata late last
night after some passengers complained of smoke in the cabin.

Figure 3.4: Stanford Dependency Parser Output

- **Data Design for dependency parser**

The details for data design for dependency parser is as follows:

Data Structure Used: Since the Stanford Dependency Parser is used to parse the sentences, we are not explicitly keeping the grammar rules used for parsing. However, 52 grammatical relations which Stanford parser utilises in its final representation are identified and taken in to account while finding concepts etc. in the next phase. All 52 relations used are given in Table 3.2.

Table 3.2: Stanford Dependency Relations

Sr. No.	Dependency Relation	Definition
1	root	Root
2	dep	Dependent
3	aux	Auxiliary
4	Auxpass	passive auxiliary
5	cop	Copula
6	arg	Argument
7	agent	Agent
8	comp	Complement
9	acomp	adjectival complement
10	ccomp	clausal complement with internal subject
11	xcomp	clausal complement with external subject
12	obj	Object
13	Dobj	direct object

14	iobj	indirect object
15	Pobj	object of preposition
16	Subj	Subject
17	nsubj	nominal subject
18	nsubjpass	passive nominal subject
19	csubj	clausal subject
20	csubjpass	passive clausal subject
21	cc	Coordination
22	conj	Conjunct
23	expl	expletive (expletive “there”)
24	mod	Modifier
25	amod	adjectival modifier
26	appos	appositional modifier
27	advcl	adverbial clause modifier
28	det	Determiner
29	predet	Predeterminer
30	preconj	Preconjunct
31	vmod	reduced, non-finite verbal modifier
32	mwe	multi-word expression modifier
33	mark	marker (word introducing an advcl or ccomp
34	advmod	adverbial modifier
35	neg	negation modifier
36	rcmod	relative clause modifier
37	quantmod	quantifier modifier
38	nn	noun compound modifier
39	npadvmod	noun phrase adverbial modifier
40	tmod	temporal modifier
41	num	numeric modifier
42	number	element of compound number
43	prep	prepositional modifier
44	poss	possession modifier
45	possessive	possessive modifier (’s)
46	prt	phrasal verb particle
47	parataxis	parataxis
48	goeswith	goeswith
49	punct	punct
50	ref	ref
51	sdep	sdep
52	xsubj	xsubj

Output Data Format: The sentence when parsed through the dependency parser has following format as shown in Figure 3.5. Here all the words are numbered and are given dependency tags in pairs.

```
root ( ROOT-0 , brought-8 ) det ( flight-4 , An-1 ) compound ( flight-4 , Air-2 )
compound   (f light-4 , India-3 ) nsubjpass ( brought-8 , flight-4 ) case ( HongKong-6
, to-5 ) nmod:to ( flight-4 , HongKong-6 ) auxpass ( brought-8 , was-7 )
compound:prt (  rought-8 , down-9 ) case ( Kolkata-11 , at-10 ) nmod:at ( brought-8 ,
Kolkata-11 ) advmod ( night-14 , late-12 ) amod ( night-14 , last-13 ) nmod:tmod (
brought-8 ,  ight-14 ) mark ( complained-18 , after-15 ) det ( passengers-17 , some-16
) nsubj (  omplained-18 , passengers-17 ) advcl ( brought-8 , complained-18 ) case (
smoke- 0 , of-19 ) nmod:of ( complained-18 , smoke-20 ) case ( cabin-23 , in-21 )det (
cabin-23 , the-22 ) nmod:in ( complained-18 , cabin-23
```

Figure 3.5: Dependency Tagged Sentence

Output Data Storage: This tagged structure of the sentence is kept into primary memory in a *String* type data after removing the brackets, hyphens and numeric values provided by the tagger.

Data Utilization This data is utilized by Information Extractor module.

- **Algorithm for Dependency Parser**

The algorithm for dependency parser is shown in Figure 3.6 as follows:

```
Algorithm dependency_parser()
Input: anaphora resolved text document
Output: text documents with dependency tags of each word
Begin
        for each sentence of the text document
        call Stanford Dependency parser
        end for
End
```

Figure 3.6: Algorithm for Dependency Parser

The algorithm takes the anaphora resolved pre-processed text document as the input to the Stanford Dependency Parser which parses the text and provides dependency tags to each word of each sentence of the document.

xx) NER Tagging

The nouns in the sentence are named entities. A noun in a sentence can refer to some person, location, organization or time. A natural language processing tool SENNA [56] is utilized here. SENNA is used in our work to provide NER tags to the nouns of the sentence.

- **Data Design for NER Tagger**

The details for data design for NER Tagger is as follows:

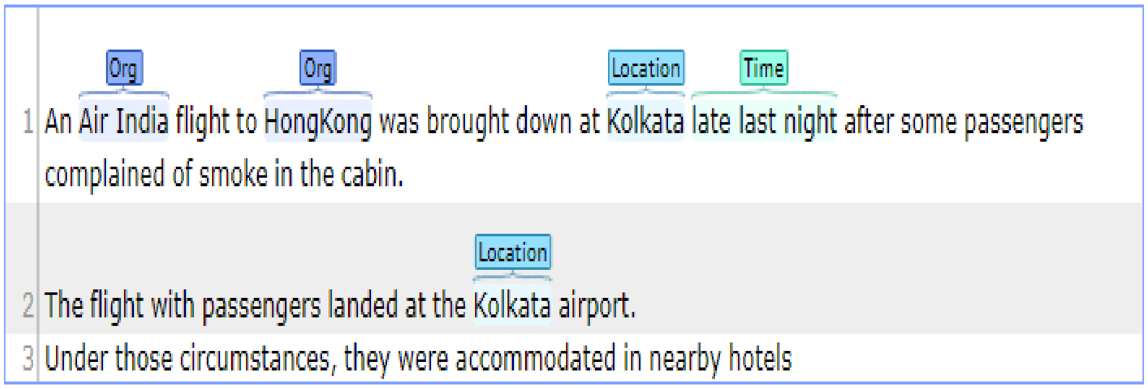
Data Structure Used: SENNA has dictionary of 130 thousand words which is used to map every word to a vector of 50 floating numbers. SENNA also keeps Gazetteer list and uppercase information as mentioned earlier in Chapter 2. SENNA provides the following tags that can be extracted by it as shown in Table 3.3.

Table 3.3: NER Tags

NER Tag
Person
Location
Organization
Time

Output Data Format: For the sentences of the example given above the following will be the NER tagged sentence shown pictorially in Figure 3.7.

S-PER, S-LOC, S-ORG and S-TIME tags may be given to the entities, B-NP,I-NP and E-NP specify the beginning, intermediate and end of the named entity.



1 An Air India flight to HongKong was brought down at Kolkata late last night after some passengers complained of smoke in the cabin.

2 The flight with passengers landed at the Kolkata airport.

3 Under those circumstances, they were accommodated in nearby hotels

Figure 3.7: NER Tagging of Sentence (Pictorial Representation)

We get the sentences along with their tags associated with each word as shown in the Figure 3.8**.**

Word	**Tag**
An	B-ORG
Air	I-ORG
India	E-ORG
Flight	O
To	O
HongKong	S-LOC
was	O
brought	O
down	O
at	O
Kolkata	S-LOC
late	S- TIME
last	S- TIME
night	S- TIME
after	O
some	O
passengers	O
complained	O
of	O
smoke	O
in	O

Figure 3.8: NER Tagged Sentence

This structure of sentences shows the respective named entity tagged for individual words such as S-LOC for location, ORG for organisation; S-TIME for time and O stands for others.

Data Storage The named entity tagged sentences are stored in a *String* in primary memory

Data Utilization: These tagged sentences can be used further by information extractor module.

- **Algorithm Design for NER Tagger**

  The algorithm for NER tagger shown in Figure 3.9 is as follows:

```
Algorithm ner_tagger()
Input:  pre-processed text document
Output: named entity tagged text document
Begin
        for each sentence of the text document
          call SENNA for name entity tagging
        end for
End
```

Figure 3.9: Algorithm for NER Tagger

The algorithm starts with passing the pre-processed document to SENNA which performs the tagging for each sentence of the document and gives the name entity tagged sentences of the document as output which are used by Information Extractor module as an input for further processing.

3.4.3 Information Extractor

The Information Extractor is designed here for extracting information from sentence structures. The information extracted is generally concepts, their semantic roles in the sentence, properties and relations. It takes the co-reference resolved sentences having dependency tags and NER tags attached to them.

i) Concept, Relation and Role Extractor

These are the concepts participating directly in a relation given by the verb in the sentence and concepts which are not directly related to the verb. Concepts may be existing in the text in the form of a single word or a multi word i.e. a single noun may be there representing the concept or a noun phrase is used in the sentence for a concept. Stanford parser takes into account of these multi word concepts very efficiently but to give these concepts a proper representation, these are combined to form a multiword concept. We analysed the dependency tags along with NER tags from SENNA and established whether the concept is performing an action or an action is being acted upon that concept. We further assigned semantic roles to these concepts depending upon whether they are actors or acted upon concepts and their NER tags e.g. a word is having dependency tag as *nsubj* and is having named entity tag as *Person*. This word will be framed as *actor concept* and the semantic role of this *actor concept* will be established as *agent*. If the dependency tag for a word is *Tmod* and the name entity tag is *Time* the concept will be having type *acted upon* and the semantic role *Temporal*. Table 3.4 shows the possible types of concepts and possible roles according to the NER and dependency tags.

Table 3.4: Dependency Tags, Concepts, Possible Roles and Relations

Dependency Tag	**Concept**	**NER tag**	**Role**	**Relation**
Nsubj Nsubjpass	actor concept	None Person Organization Location	Agent agent organization location	Verb
Rcmod Vmod	actor concept	None	Agent	Verb
Dobj pobj iobj xcomp agent	acted upon concept	None	Theme	Verb

Tmod	acted upon concept	Time	Temporal	Verb
Prep_ X	acted upon concept/indirect concept			
X=with		none	accompanier	
X=to		none		
		location	location	
X=in		none	theme	
		location	location	
X=at		time	temporal	
		none	theme	
X=after		location	location	
		time	temporal	
X=on		none	theme	
Poss	actor concept acted upon concept	None None	Agent Possession	Has

- **Data Design for concept_relation_role extractor**

The information thus extracted using Table 3.4 by this module needs to be stored in tables in a database. Following tables are created for this module:

i) Concept Table

The concept table contains the all the concepts extracted from all the documents with their respective document_id, sentence_ id and concept_ id assigned to each concept. This table is filled by the concept_extractor module and is stored in secondary memory.

Data Format: The metadata for concept table in Java DB (Derby) database is as shown in Table 3.5.

Table 3.5: Metadata for Concept Table

Field Name	**Data Type**
Doc_id	VARCHAR
Sentence_id	VARCHAR
Concept_id	VARCHAR
Concept_Name	VARCHAR

The metadata for Concept Table contains Document identification number Doc_id, Sentence identification number sentence_id, Concept identification number Concept_id and the name of the concept Concept_name.

Data Storage: All these fields are stored as having *VARCHAR* as their datatype.

Data Utilization: Concept table is utilized by Property_extractor and Hierarchy_extractor modules by bringing it into primary memory.

ii) Object_Property_Table

This table contains the extracted actor concepts denoting the subjects, Acted upon concepts denoting the objects, their semantic roles in the sentence and the relation between the subject and the object denoted by relation_name.

Data Format: The metadata for Object_Property_Table in Java DB is shown in Table 3.6.

Table 3.6: Metadata for Object_Property_Table

Field Name	**Data Type**
Doc_id	VARCHAR
Sentence_id	VARCHAR
Rel_id	VARCHAR
Actor_concept_id	VARCHAR
Ac_has_role	VARCHAR
Relation_Name	VARCHAR
Acted_upon_concept	VARCHAR
Au_has_role	VARCHAR

The metadata for Object Property Table contains Document identification number Doc_id, Sentence identification number sentence_id, Relation identification number, Actor Concept identification number Actor_concept_id, the semantic role of actor concept, Acted upon Concept identification number Acted_upon_concept_id, the semantic role of acted upon concept, and the name of the object propertyrelation _name.

Data Storage: All these fields are stored as having *VARCHAR* as their datatype.

Data Utilization: This table is filled by object_property module and is stored in secondary memory. This table is drawn into primary memory so that it can be used by ontology_matcher_merger module.

- **Algorithm Design for concept_relation_role_extractor()**

Concept extraction, relation extraction and role extraction go hand in hand. The concepts on which the action is being done and concepts doing action are captured along with the action i.e. the relation among these concepts and are stored in tables having concepts and relation associated with them. The algorithm for extracting concepts, their semantic role in the sentence and relations among the concepts takes the sentence token list, dependency tags and name entity tags from the output the natural language processing module. It also uses Table 3.4 which maps the concepts to their semantic roles according to their dependency tag and name entity tag. The algorithm is given in Figure 3.10.

Here in this algorithm, the output of natural language processing module i.e. the sentence token list which is dependency parser tagged and the name entity tags from SENNA are given as input. Depending upon the dependency tags and name entity tags, concepts are extracted in step 1.The concepts are generalized as actor concepts or acted upon concepts in step 1.a and step1.b and provided their semantic roles according to their tags and Table 3.4 in step 1.e. The algorithm also looks for the multiword concepts and extracts and stores them by adjoining them with an underscore in step 2. Object properties (relations among these concepts) are also extracted by this algorithm in step 1.e. The concepts along with their semantic roles and the relations between them are stored in Object_Prop_Role_Table in step 3.

```
Algorithm concept_relation_role_extractor()
Input: sentence token list, dependency tags, name entity tags from SENNA
Output: actor concepts, acted upon concepts, roles, relations
Begin
  for each token in the sentence list
  1.  analyze the dependency tag and name entity tag for possible concepts, relations
  and role                //refer columns Dependency tag and Concept from Table
  3.4.
  a) extract actor concepts
  b) extract acted upon concepts
  c) extract concepts not directly related to verb
                                          //refer columns NER tag and Role from Table
  3.4.
  d) extract roles
                                         //refer column Relation from Table 3.4.
  e) extract relations
                                        //refer column Dependency tag  from Table 3.4
  2. analyze the dependency tag  for  "nn" or "nnp"
                                          // multiword concepts representation
       a)  concatenate the concepts using "_"
       b) store all concepts in concept_table
  3. store concepts, roles, relations in  Object_Prop_Role_ Table
  end for
  End
```

Figure 3.10: Algorithm for Concepts, Roles and Relation Extractor

ii) Property Extractor

The properties of the concepts participating in ontology may have some property associated with them given by the sentence. These properties are those that modify the concept in some manner .e.g. in the following sentence:

“This is a red book”.

red which is an adjective in the sentence becomes data property for the concept *book* in the ontology. These properties are extracted by processing the dependency parsed structure of the sentence.

- **Data Design for property extractor**

These extracted properties are stored along with their concepts in a table in the database.

Data Format: We use data property table that contains the properties of the extracted concepts. The metadata for Data_Property table in Java DB is given as follows in Table 3.7.

Table 3.7: Metadata for Data_Property Table

Field Name	**Data type**
Doc_id	VARCHAR
Sentence_id	VARCHAR
Concept_id	VARCHAR
Has_prop	VARCHAR

The metadata for Data_Property Table contains the Document identification number Doc_id, Sentence identification number Sentence_id, Concept identification number Concept_id and a property field has_prop to show the data property associated with the concept.

Data Storage: This table is stored in secondary memory.

Data Utilization: This table is brought into primary memory to be used by Ontology Generator and Ontology Enricher (Chapter 4).

- **Algorithm Design for property_extractor**

The properties that modify concepts in some manner are called data properties and are extracted from text according to their dependency tags given as input along with the sentence token list. The algorithm for extracting properties is given in Figure 3.11.

```
Algorithm property_extractor()
Input: sentence tokens list with dependency tags next to each token in the list
Output: concepts properties
Begin
    1. for each token in the sentence list
      2. for each concept  from concept_ list
        a) analyze the dependency tag "amod" for extracting properties of concepts
        b) store the concept properties in data_prop_table
      end for
```

Figure 3.11:Algorithm for Property Extractor

In this algorithm, the output of natural language processing module is given as input which is sentence token list along with their dependency tags. In step 1 and step 2.a, this algorithm analyses the tokens of each sentence having '*amod*' dependency tag which identify the adjective associated with the concept determining its data property. These data properties are stored in a Data_Property Table along with their associated concept in step 2.b.

iii) Hierarchy Extractor

From the list of concepts extracted in Concept_Table hierarchy of concepts is extracted. All the concepts from Concept_Table are brought to a concept list in primary memory. The concepts are syntactically matched to determine whether some concept matches partially with some other multiword concept (compound nouns). If a concept matches with head noun of the compound noun, it is added to the hierarchy table with relation "has_a". Otherwise if a concept matches with other part of compound noun which is not head noun, it is added to hierarchy table with relation "has_ instance".

- **Data Design for Hierarchy Extractor**

Hierarchy table is generated if concepts are related to each other with part-whole relation or is-a relation. This table captures the possible taxonomical relations among concepts.

Data Format: The metadata for Hierarchy_Table is shown in Table 3.8 as follows:

Table 3.8: Metadata for Hierarchy Table

Field Name	**Data type**
Doc_id	VARCHAR
H_id	VARCHAR
Sub_concept	VARCHAR
Super_concept	VARCHAR

The metadata of this table contains the document identification number Doc_id, Hierarchy identification number H_id, name of the sub concept sub_concept and the name of super concept super_concept.

Data Storage: This table is generated by Hierarchy_extractor module using the concept table and is stored in secondary memory.

Data Utilization: This table is utilized by ontology_matcher_merger by bringing it into primary memory so that concepts having a hierarchy in the form of sub concepts and super concepts can be merged.

- **Algorithm for Hierarchy Extractor**

There may be a possibility that some concepts extracted from text possess a hierarchical relation to some other concepts. This relation can be an *is-a* relation or *has-instance* relation. The hierarchy extractor algorithm takes the concept list and finds the super concepts and their sub concepts and establishes the proper relation between them. The algorithm is given as in Figure 3.12.

The algorithm takes the list of concepts which include single word and multiword concepts and generalises them into a hierarchy if exist. The single word concept may be subsumed by some other multiword concept. The single word concept may be same as the head noun of the multiword concept or as the other part of the multiword concepts. This algorithm stores both of these concepts by establishing has_instance or has_a relation between them in hierarchy_Table..

```
Algorithm hierarchy_extractor()
Input:  concept_ list from concept_Table
Output: super_ concept, sub_ concept
 Begin
   1. for each concept from concept_ list
        compare concept with each multi word concept syntactically  for partial matching
        a)  If concept matches with head noun of multi word concept
              super_concept = concept
              Relation = "has_a"
              sub_concept= multiword concept
        b)  If concept matches with other part of multi word concept except head noun
              super_concept = concept
              Relation = "has_instance"
              sub_concept= multiword concept
   2.  store in Hierarchy_Table
      end for
 End
```

Figure 3.12: Algorithm for Hierarchy Extractor

3.4.4 Sub-Ontology Generator

For each sentence a sub-ontology is created using Jena API comprising of concepts along with their semantic roles, data properties, relations as their object properties, and hierarchy if exists. These sub-ontologies are shown graphically by using the GraphViz tool [57].

- **Data Design for sub-ontology generator**

The contents of Object_Property_Role_Table, Properties_Table and Hierarchy_Table are brought into lists in the primary memory. These lists are used to create sub-ontology for each sentence using Jena API.

Data Format and Storage: The sub-ontologies are stored in *rdf* structures in the memory**.**

Data Utilization: These sub-ontologies are used further by sub-ontology merger to construct the final ontology.

3.4.5 Sub-Ontology Merger

The sub-ontologies thus created for each sentence are merged to form a single ontology. This process starts with merging the very first ontology with a NULL ontology. For merging ontologies relations and concepts of different ontologies are matched for their syntactic and semantic similarity using WordNet [60] and the Hierarchy_Table. The semantic roles of concept nodes are also used in the matching process. These are also matched for similarity wherever there are similar concepts..

This module considers all cases where relation or concept nodes of an ontology may or may not be matching to concept or relation node of other ontology. Also there may be dissimilarity in matching concept nodes.

- **Data Design for sub-ontology merger**

To tackle these cases we have purposed some rules here that are stored in primary memory according to which nodes in sub-ontology are merged with their corresponding semantic similar nodes of other ontologies.

Data Structure Used: Following rules are designed in our work which are being applied in the algorithm for merging sub-ontologies

Rule 1: Relations in both sub ontologies are semantically same. Also their respective concepts and the roles of concepts in these relations are matching semantically. In addition to this hierarchies of concepts are taken into consideration for merging the concepts as shown in Figure 3.13.

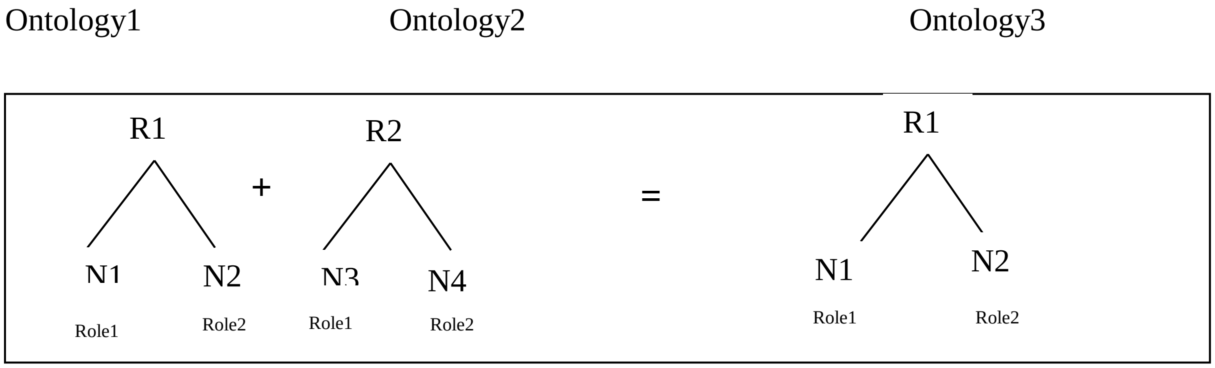


Figure 3.13: Rule 1

Rule 2: The dangling non-matching concept node in sub-ontology. In this case relations R1 of Ontology1 and R2 of Ontology2 are semantically similar and their related concept nodes i.e. N1 is semantically matched with N3 and N2 is semantically matched with N4. Here N5 is a non-matched concept, which will be aligned as shown in Figure 3.14.

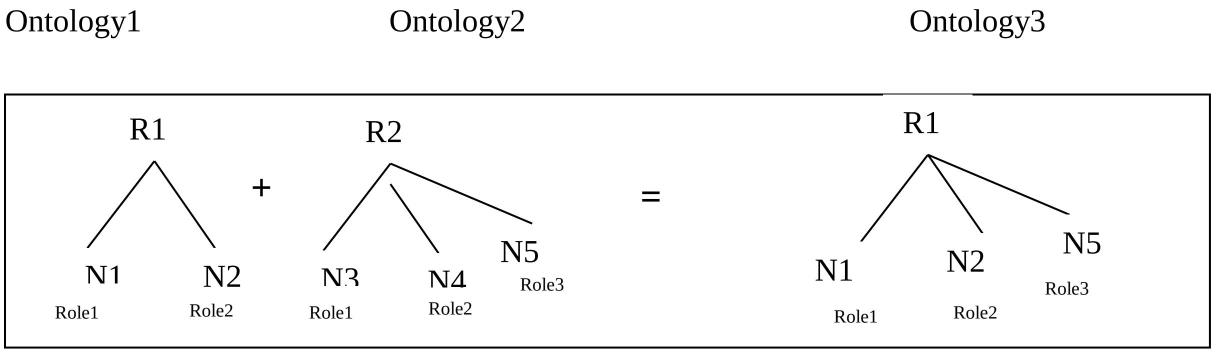


Figure 3.14: Rule 2

Rule 3: Some concept nodes are semantically matching, but corresponding relation is semantically dissimilar in both sub-ontologies. In this case relations R1 of ontology1 and

R2 of ontology2 do not match while the concept N1 and N3 are similar. These will be aligned as shown I Figure 3.15.

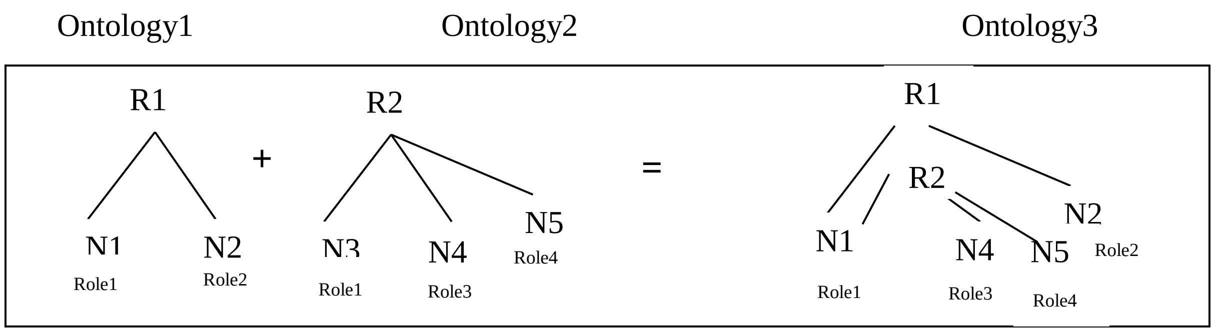


Figure 3.15: Rule 3

Rule 4: Concept nodes are matching semantically excluding their semantic roles and their corresponding relations are also semantically dissimilar in both ontologies. The matching concepts are merged to be a single concept keeping their respective semantic roles in each sentence intact. In this case relations R1 of ontology1 and R2 of ontology2 do not match while the concept N1 and N3 are semantically similar having roles Role1 and Role3 respectively. These will be aligned as shown in Figure 3.16.

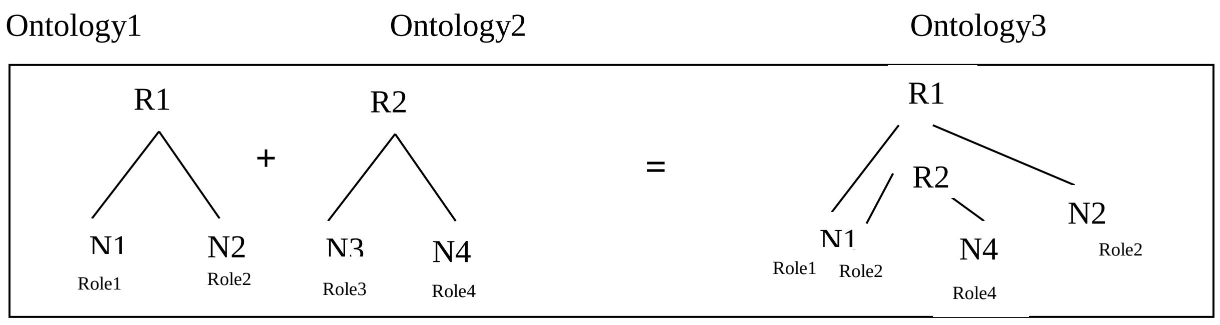


Figure 3.16: Rule 4

Data Utilization: These rules are used by ontology matcher merger to construct the final ontology.

- **Algorithm for ontology matcher merger**

In the process of ontology construction here sub ontologies are created for each sentence after extracting the concepts, their semantic roles and establishing the correct hierarchy among them and extracting their object and data properties. These sub-ontologies are matched and merged to form the whole ontology. Some rules have been crafted for the purpose of matching and merging these sub-ontologies which are utilized by the ontology_matcher_merger algorithm which takes two different sentence sub-ontologies as input, uses WordNet in the process of matching and merges the ontologies according to these rules to produce a merged ontology. The algorithm for matching and merging ontologies is given as follows in Figure 3.17.

```
Algorithm ontology_matcher_merger()
Input: Ontology O1, Ontology O2, WordNet
Output: Merged ontology O3
Begin
1. Check if O1 is NULL then assign O3= O2;
2. Analyze relation in both ontologies for equality
a) analyze actor concepts and acted upon concepts of both ontologies for equality
// Rule 1  (Figure 3.13)
b) analyze actor concepts role and acted upon concepts role of both ontologies for
   equality
3. integrate the two ontologies by merging the sub-ontologies where the matching is
   found and store in O3.
4. align the non-matching dangling concepts // Rule 2 (Figure 3.14)
5. align non matching relations pairs with non-matching concepts    // Rule 3 (Figure
   3.15)
6. align non matching relations pairs with matching concepts    // Rule 4 (Figure 3.16)
End
```

Figure 3.17: Algorithm for Ontology Matcher Merger

This algorithm applies the devised rules by us for merging the ontologies in a manner such that all the concept pairs, relation pairs are aligned while integrating these ontologies. In step 1, there is a check to ascertain that any of the ontologies to be merged are not null otherwise the merged ontology will be the not null ontology itself. Then in step 2 and 3, the relation pairs from both ontologies are analysed to find the matching concept pairs or matching relation names. The rule devised are applied here in step 4, 5 and 6 to integrate these ontologies and assure that all the matching concept pairs or matching relation pairs are merged well and the non-matching relations or non-matching concepts are aligned well in the integrated ontology. Ontology Representer

The sentence ontologies are merged in the ontology matcher merger and regenerated to form complete text document ontology using Jena API. The merged final ontology can be converted into dot file which is further shown graphically by GraphViz tool [57].

iv) Data Design for ontology representer

The document ontologies are stored in secondary memory in .rdf or .owl form.

The above described algorithms of different modules are used by ontology generator algorithm. This algorithm takes plain text documents as input. The process for generating complete ontology from a set of documents is given in the algorithm as shown in Figure 3.18.

- **Algorithm Design of Ontology_ Generator**

This algorithm generates the overall ontology for a given text document by calling the methods defined for the all the modules of the system as shown in Figure 3.18. The algorithm takes the unstructured text document as input and processes it sequentially through these methods and gives a final ontology of that text document.

```
Algorithm Ontology_ Generator
Software Tools Used: Stanford Dependency Parser, Senna, JavaRAP,
Input: A set of text documents,  WordNet
Output: Ontology of documents
Begin
for each document
      for each sentence
    1. pre_processor();
    2. anaphora_resolver();
                                        //output is pronouns resolved to their
                                                            noun mentions
    3. dependency_parser();
                              //output is part of speech tagged sentences along with
                                                        //their dependencies
    4. ner_tagger();

                                   // output is named entity tagged nouns
    5. concept_relation_role_extractor();
                                   //output is concept, roles, object properties

    6. property_extractor();             //output is data properties of concepts

    7. hierarchy_extractor();              // output is hierarchies of concepts if
       exist
    8. Generate sentence ontologies using Jena API for ontology
                                        // generate sentence   ontologies
    end for

    9. ontology_matcher_merger();   // match and merge ontologies to generate full
       ontology
    10. Generate the ontology in .rdf  format using Jena API and GraphViz
end for
End
```

Figure 3.18: Algorithm for Ontology Generator

The algorithm first starts with processing each sentence of each document. Each sentence is preprocessed by the module pre_processer to remove transition words or phrases and then anaphora_resolver() is called in step 2 which resolves the coreferences. Dependency tagging is carried out in step 3 followed by named entity recognition in step 4. After performing all the natural language processing on the document, this is passed for further processing to the Information Extractor. In step 5 concepts, relations and semantic roles are extracted by calling concept_relation_role_extractor().

In step 6 data properties of the concepts are extracted by calling property_extractor. In step 7 hierarchies are established by calling hierarchy_extractor(). Step 8 generates the sentence ontologies. In step 9, these ontologies are merged to form the final ontology by calling ontology_matcher_merger(). Step 10 generates and shows the final ontology using Jena API and also shows the ontology graphically through GraphViz tool.

3.5 WORKING OF SYSTEM THROUGH EXAMPLE

The process of constructing ontology takes place in two steps. First step is to form the sub-ontological structures from the sentences of the document and in second step the full document ontology is built using the previously formed sentence sub-ontologies. It is assumed here that the inputs are correct.

For a sample text document having following three sentences:

Sentence1: *An Air India flight to HongKong was brought down at Kolkata late last night after some passengers complained of smoke in the cabin.*
Sentence 2: *The flight with passengers landed at the Kolkata airport.*
Sentence 3: *Under those circumstances, Passengers were accommodated in nearby hotels*

After processing the sentences from all the extractors except following tables are obtained. The Concept_Table contains the concepts retrieved from all sentences in all documents as shown in Table 3.9.

Table 3.9: Concept_Table

Doc_id	**Sentence_id**	**Concept_id**	**Concept_Name**
D0	S0	Cid0	Air_India_Flight
D0	S0	Cid1	HongKong
D0	S0	Cid2	Kolkata

D0	S0	Cid3	Passengers
D0	S0	Cid4	Smoke
D0	S0	Cid5	Cabin
D0	S1	Cid6	Flight
D0	S1	Cid7	Passengers
D0	S1	Cid8	Kolkata_airport
D0	S2	Cid9	Passengers
D0	S2	Cid10	Night
D0	S2	Cid11	Hotels

Table 3.10 contains the object properties i.e. relations among the concepts and role each concept is playing. As mentioned above some concepts are directly related to the verb in the sentence. These concepts are stored in table with the relation name.

Other concepts which are not directly related to the verb directly are stored without relation name and their relation is identified by the name of label and role each concept is playing in relation with the other concept.

Table 3.10: Object_Property_Role_Table

Doc_id	**Sentence_id**	**Rel_id**	**Actor_concept_id**	**Ac_has_role**	**Relation_Name**	**Acted_upon_concept_id**	**Au_has_role**
D0	S0	Rel_id0	Cid0	Agent	Brought_down	Cid2	Location
D0	S0	Rel_id1	Cid3	Agent	Complained	Cid4	Theme
D0	S1	Rel_id2	Cid6	Agent	Landed	Cid8	Location
D0	S2	Rel_id3	Cid9	Agent	Accommodated	Cid11	Theme
D0	S0	Rel_id0	Cid10	Temporal	Brought_down	Cid0	Agent
D0	S0	Rel_id1	Cid5	Location	Complained	Cid3	Agent
D0	S0	Rel_id4	Cid1	Location	To	Cid0	Agent
D0	S1	Rel_id5	Cid7	Accompanier	With	Cid7	Agent

Table 3.11 stores the data properties of the concepts extracted from each sentence in the document by the property extractor module.

Table 3.11:Property_Table

Doc_id	**Sentence_id**	**Prop_id**	**Concept_id**	**Has_prop**
D0	S1	Pid0	Night	Last
D0	S2	Pid1	Hotels	Nearby

Hierarchy extractor module finds the hierarchy among the concepts from the Concept_ Table and stores in Hierarhcy_Table as shown in Table 3.12.

Table 3.12:Hierarchy_Table

Doc_id	**H_id**	**Sub_concept**	**Super_concept**
D0	H_id1	Air_india_flight	Flight
D0	H_id2	Kolkata_airport	Kolkata

For each of these three sentences, ontologies are generated in the first step (Figure 3.19).

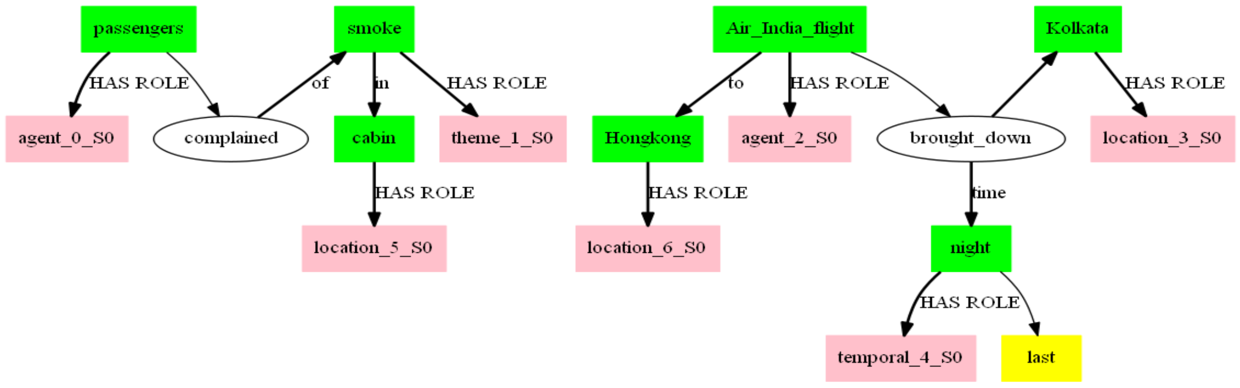
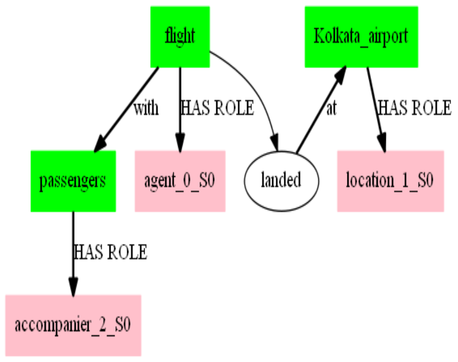
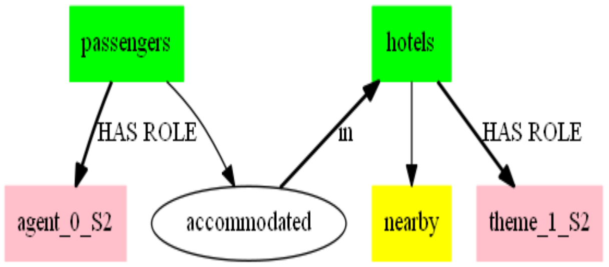


Figure 3.19: Sentence Sub-ontologies (Cont. on next page)





Here green boxes show the concepts, ellipses show the object properties, yellow boxes show the data properties and the pink boxes show the roles of the concepts. The second step here is to apply ontology matching and merging algorithm on these ontologies. Here relations (object properties) as shown by the ellipses in the sentence ontologies are considered at first. All possible base forms from WordNet [60] are taken for each relation and used for matching similarity. Table 3.13 shows the possible base forms of the relations identified in the document.

Table 3.13: Base Forms of Relational Words

Relation Name	**Possible base form of the word in WordNet**
brought down	lower, take down, let down, get down, bring down: move something or somebody to a lower position, overthrow, subvert, overturn, cause the downfall of, impose something unpleasant, **land**, put down, cause to come to the ground, reduce, cut down, cut back, trim, trim down, trim back, cut, cut down on; make a reduction in
Landed	set down: reach or come to rest, put down, bring down: cause to come to the ground, **bring down**, bring into a different state, bring ashore, deliver (a blow), set ashore, shore: arrive on shore, shoot down, land: shoot at and force to come down, landed: owning or consisting of land or real estate
Accommodate	suit, fit: be agreeable or acceptable to, adapt, make fit for, or change to suit a new purpose, provide with something desired or needed, hold, admit: have room for; hold without crowding, lodge, accommodate: provide housing for, oblige, accommodate: provide a service or favour for someone, reconcile, conciliate: make (one thing) compatible with (another)
Complain	kick, plain, sound off, quetch, kvetch: express complaints, discontent, displeasure, or unhappiness, make a formal accusation; bring a formal charge

There can be different cases while applying the ontology_matcher_merger algorithm. Each case is considered separately and the purposed rules are applied to match and merge different constituents of the ontology.

Case 1 Applying the Rule 1(Figure 3.13)

Each base form of the word or phrase representing the relation in one sentence is matched with -the all possible base forms of the relations in other sentences for applying the Rule 1. As evident from the above table the relations *brought_down* and *landed* match as the relation *brought_down* has base form *land* and also *landed* has base form *bring down* so *borught-down* in ontology1 replaces *landed* in the merged ontology. *Air_india_flight* and *Fligh*t are matched using hierarchy_Table and their respective roles are also matched. In the similar manner *Kolkata* and *Kolkata_airport* are matched using Hierarchy_Table and by matching their respective semantic roles as shown in Figure 3.20.

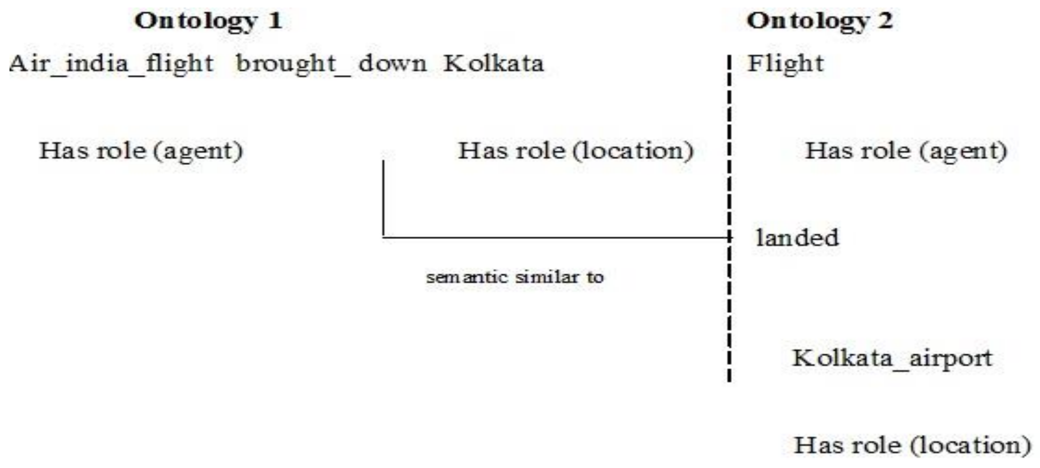
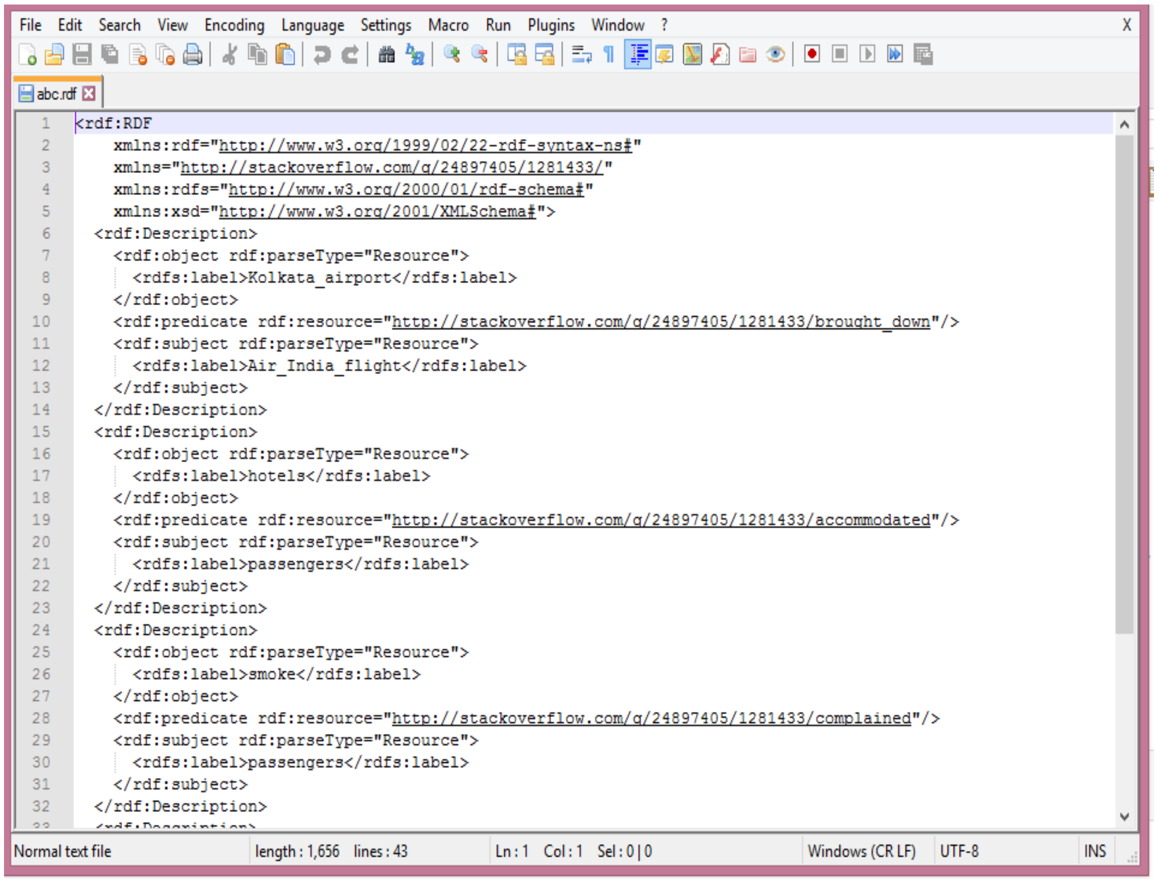


Figure 3.20: Similarity Matching in Case 1

C**ase 2** *HongKong* is a dangling non matching concept in Ontology1 and is merged with Ontology2 according to Rule2 (Figure 3.14).

Case 3 The concept "*passengers*" has role "*agent*" in first ontology and is in relation with "*complained*" while in second ontology it's role is "*accompanier_with*" with no matching relation in Ontology2. In the merged ontology there is just one node for "*passengers*" and has three roles *agent_s0*, *accompanier_with_s1*, and *agent_s2* which is obtained by applying Rule4 (Figure 3.16) again while merging this with Ontology3.

The *rdf* file generated corresponding to this sample document is shown in Figure 3.21.


File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
abc.rdf

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://stackoverflow.com/q/24897405/1281433/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <rdf:Description>
    <rdf:object rdf:parseType="Resource">
      <rdfs:label>Kolkata_airport</rdfs:label>
    </rdf:object>
    <rdf:predicate rdf:resource="http://stackoverflow.com/q/24897405/1281433/brought_down"/>
    <rdf:subject rdf:parseType="Resource">
      <rdfs:label>Air_India_flight</rdfs:label>
    </rdf:subject>
  </rdf:Description>
  <rdf:Description>
    <rdf:object rdf:parseType="Resource">
      <rdfs:label>hotels</rdfs:label>
    </rdf:object>
    <rdf:predicate rdf:resource="http://stackoverflow.com/q/24897405/1281433/accommodated"/>
    <rdf:subject rdf:parseType="Resource">
      <rdfs:label>passengers</rdfs:label>
    </rdf:subject>
  </rdf:Description>
  <rdf:Description>
    <rdf:object rdf:parseType="Resource">
      <rdfs:label>smoke</rdfs:label>
    </rdf:object>
    <rdf:predicate rdf:resource="http://stackoverflow.com/q/24897405/1281433/complained"/>
    <rdf:subject rdf:parseType="Resource">
      <rdfs:label>passengers</rdfs:label>
    </rdf:subject>
  </rdf:Description>
  <rdf:Description>
```

Normal text file   length : 1,656   lines : 43   Ln : 1   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS

Figure 3.21: .rdf File of Sample Document

The final ontology can be shown graphically using GraphViz in which the dot file is generated first as shown in Figure 3.22.

```
digraph mygraph{
  complained ->brought_down
{  ranksep=1.2  rankdir=LR; complained brought_down  }
  brought_down ->accommodated
{  ranksep=1.2  rankdir=LR; brought_down accommodated  }
  passengers ->complained
  passengers [shape=box,style=filled,color=Green];
  passengers -> agent_0 [style=bold,label="HAS ROLE"];
  agent_0 [shape=box,style=filled,color=Pink];
  complained ->cabin
{ rank= same rankdir=TB passengers cabin  }
  cabin [shape=box,style=filled,color=Green];
  cabin -> location_0  [style=bold , label="HAS ROLE"];
  location_0 [shape=box , style=filled , color=Pink];
  hotels ->nearby
  nearby [shape=box,style=filled,color=Yellow];
  Air_India_flight ->brought_down
  Air_India_flight [shape=box,style=filled,color=Green];
  Air_India_flight -> agent_1 [style=bold,label="HAS ROLE"];
  agent_1 [shape=box,style=filled,color=Pink];
  brought_down ->Kolkata_airport
{ rank= same rankdir=TB Air_India_flight Kolkata_airport  }
  Kolkata_airport [shape=box,style=filled,color=Green];
  Kolkata_airport -> location_1  [style=bold , label="HAS ROLE"];
  location_1 [shape=box , style=filled , color=Pink];
    agent_2 [shape=box,style=filled,color=Pink];
  brought_down ->Kolkata_airport
{ rank= same rankdir=TB Air_India_flight Kolkata_airport  }
    location_2 [shape=box , style=filled , color=Pink];
  Passengers ->accommodated
  Passengers -> agent_3 [style=bold,label="HAS ROLE"];
  agent_3 [shape=box,style=filled,color=Pink];
  accommodated ->hotels
{ rank= same rankdir=TB Passengers hotels  }
  hotels [shape=box,style=filled,color=Green];
  hotels -> location_3  [style=bold , label="HAS ROLE"];
  location_3 [shape=box , style=filled , color=Pink];
  night [shape=box,style=filled,color=Green];
  night ->brought_down[style=bold,label="time"];
  night ->temporal_0[style=bold,label="HAS ROLE"];
  temporal_0 [shape=box,style=filled,color=Pink];
  night ->last
  last [shape=box,style=filled,color=Yellow];
  Hongkong [shape=box,style=filled,color=Green];
  Hongkong ->Air_India_flight[style=bold,label="to"];
  Hongkong ->location_1[style=bold,label="HAS ROLE"];
  location_1 [shape=box,style=filled,color=Pink];
  passengers [shape=box,style=filled,color=Green];
  passengers ->Air_India_flight[style=bold,label="with"];
  passengers ->accompanier_2[style=bold,label="HAS ROLE"];
  accompanier_2 [shape=box,style=filled,color=Pink];
}
```

Figure 3.22: dot File for sample document

This dot file is then converted by the GraphViz software to a graph depicting the final merged ontology as shown in Figure 3.23.

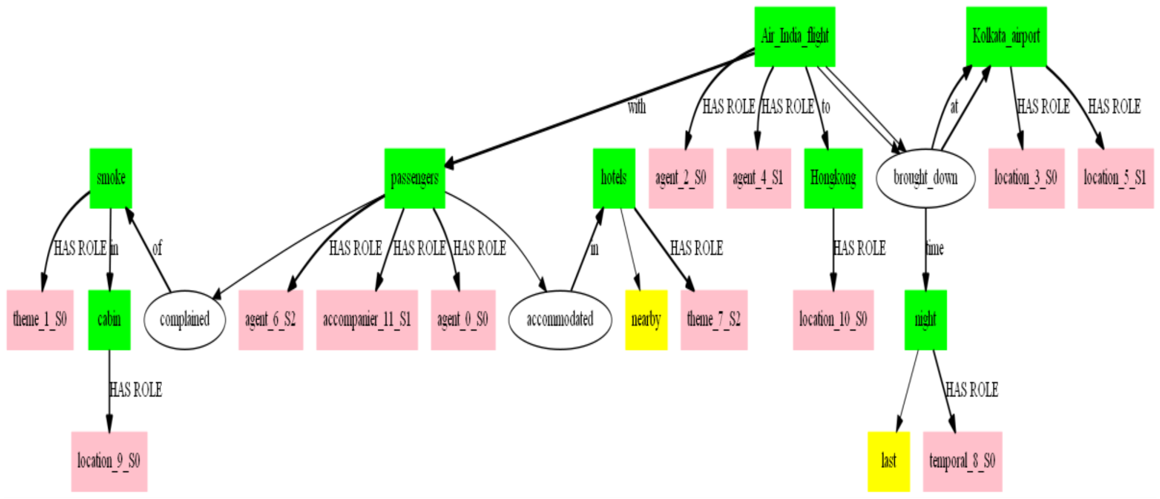


Figure 3.23: Merged Ontology of the Document

In the merged ontology redundant concepts are represented by a single node having multiple edges. Hence the merged concepts of the documents are represented in the ontology with the roles they are playing in each sentence.

3.6 IMPLEMENTATION DETAILS

We have implemented our work on Intel Core i3 with 4GB RAM using Windows 7 Operating System. We have used NetBeans 8.0.2 which is using Apache Derby as relational database which is bundled with NetBeans 8.0.2. Apache Derby is based on Java, JDBC and SQL standards. We have used Stanford Dependency Parser for annotating the sentences of text documents. The corpus on which Stanford Dependency Parser has been trained by [63] contains about 250,000 words of unedited web text. SENNA tool is used here for name entity tagging. We have used JavaRAP tool for anaphora resolution. WordNet [64] is used here for finding synonyms for matching and merging the concepts. GraphViz tool is used for showing the ontology pictorially. We have used Jena API to construct and store our ontology in the form of *.rdf* files.

Dataset-A set of 50 random news articles in English language has been taken for experimentation. These news articles are assumed to be grammatically correct.

As it is not convenient to show the ontologies for each news article in the set, implementation of a few news articles is displayed as follows:

Document1: *Prime minister Narendra Modi will leave for the Belgium capital tomorrow night. He will attend the Nuclear Security Summit in Washington and visit Saudi Arabia. Prime minister will take part in the long-pending Summit for the first time.*

The final ontology will be shown graphically in the Figure 3.24

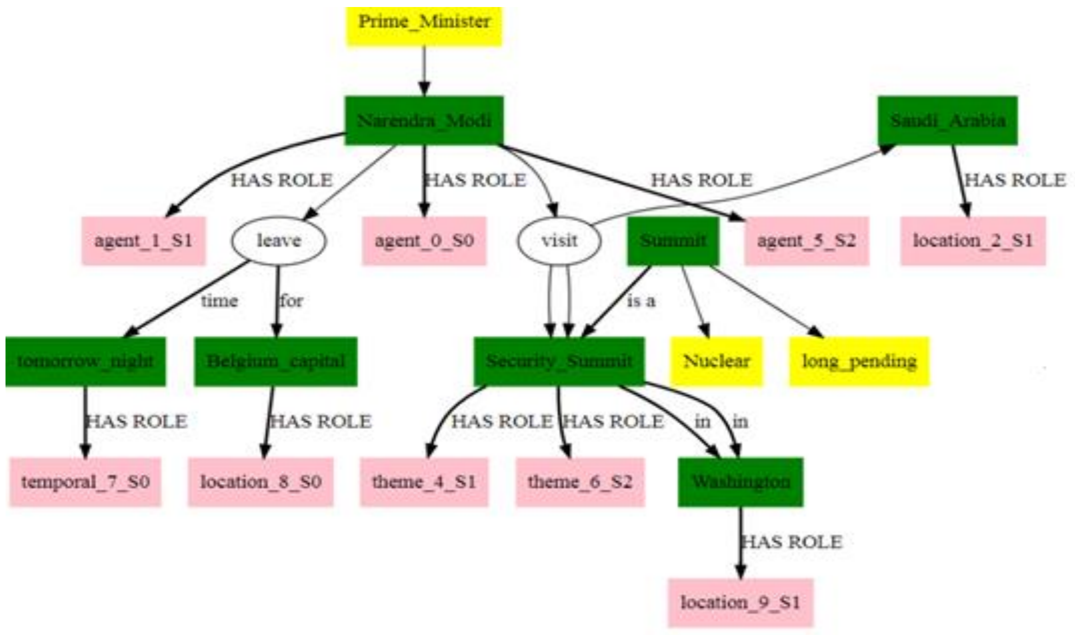


Figure 3.24: Ontology of Document 1

Document 2 : *Sania Mirza was born in Mumbai and settled at Hydrabad. She began playing tennis at early age. She became a great tennis player and she defeated top player Nadia*.

The ontology for Document 2 will be shown graphically in the Figure 3.25

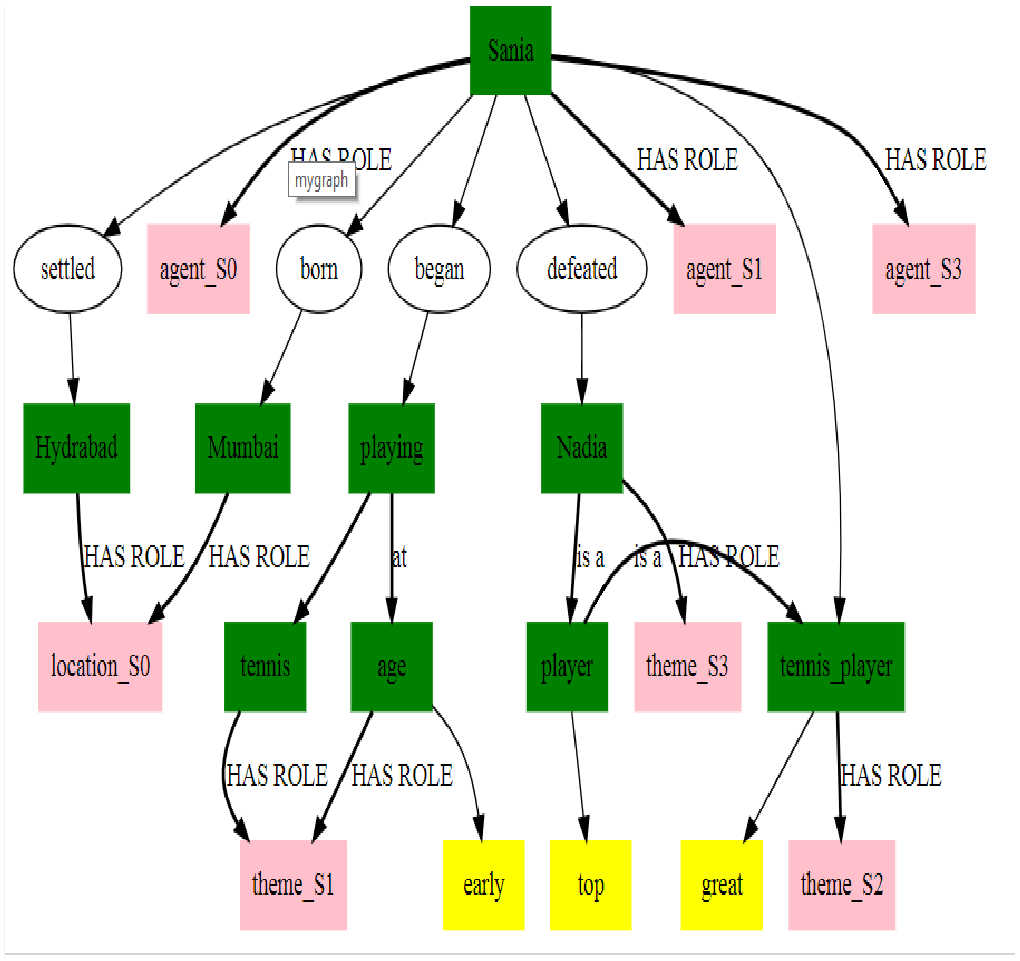


Figure 3.25: Ontology of Document 2

For **Document 3: "***The Mayor and water supply officials of Mangaluru City Corporation have been claiming that there is enough water at a vented dam at Shambhoor, on the upstream of the Thumbe dam. They are exposed as a reality check on Sunday revealed that the dam of a hydro power project at Shambhoor is empty"*. The ontology will be shown graphically in the Figure 3.26.

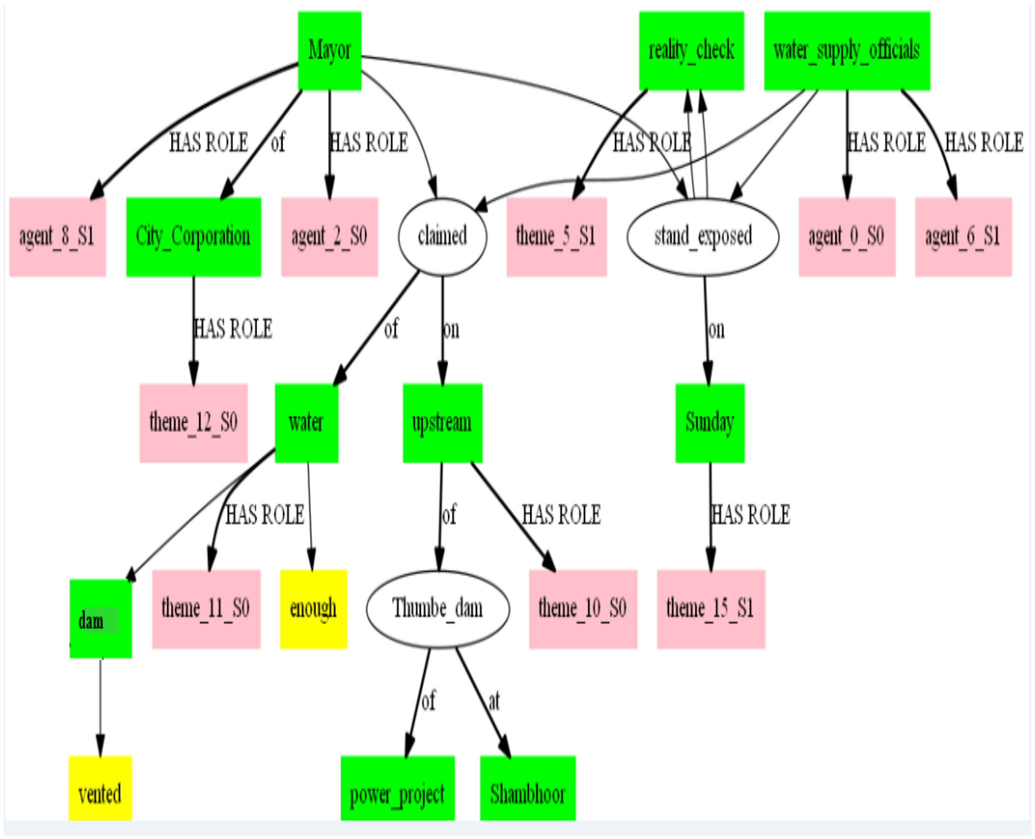


Figure 3.26: Ontology of Document 3

3.7 PERFORMANCE EVALUATION

The set of 50 news articles taken for experimentation is used also as test data set for performance analysis. We have compared our system with Open Calais [31] system by Thomson Reuter's which is linked to a market leading ontology extracting entities (persons, events, places), relationships etc and gives results in *rdf* format.

As shown in the Table 3.14, our system has scored similar precision as the other system. But our system outscores in recall and F-measures to Open Calais system in extracting the correct entities and relations.

Table 3.14: Result Comparison

System	**Precision**			**Recall**			**F-measure**		
	Concept	**Relation**	**Data Properties**	**Concept**	**Relation**	**Data Properties**	**Concept**	**Relation**	**Data Properties**
Open-Calais	100%	100%	100%	56%	25%	69.20%	71.79	40.4	81.79
Proposed System	100%	97.10%	100%	87.28%	82%	86.80%	93.2	88.9	92.9

The evaluation results indicate that our system provides good results in constructing ontology. The reason for the better performance of our system is its ability to extract all the non-taxonomical relation as compared to the other system which works on a specified set of taxonomical and non-taxonomical relations.

The reason for low recall in extracting relations in our system is that while pre-processing we remove some transition words(Table 3.1) to reduce the complexity of sentence e.g. *as was previously stated, speaking about, that being the case, to summarize* etc. The verbs in these transition words are not taken as relations in the ontology. Moreover the system is dependent on the accuracy of the parser used for extracting dependency. Stanford dependency parser has an F-score of 85.78 [65] for attaching noun phrase, modifier, clause etc.

We can show the performance analysis of our system and Open Calias graphically as shown in Figure 3.27.

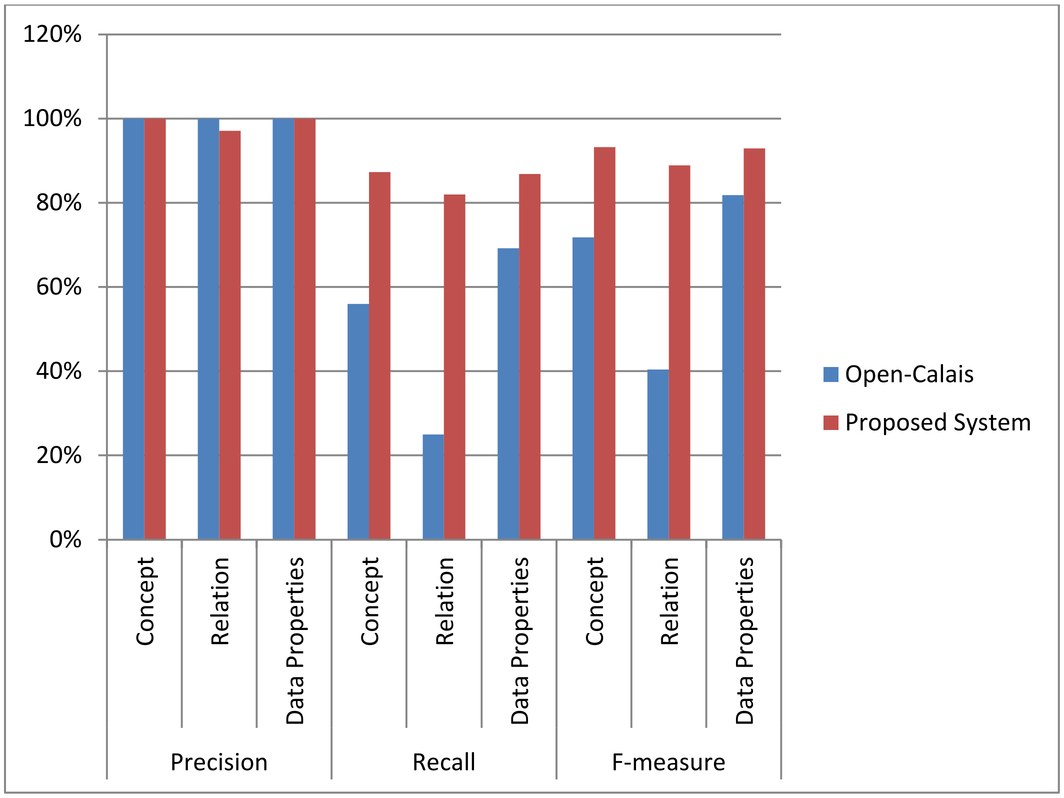


Figure 3.27: Performance Analysis of Proposed System and Open Calais System

As stated earlier our system extracts all the concepts not only the key concepts as are drawn from the text by the other system.

To improve the results with extracted relations, other ontologies or a thesaurus besides WordNet can be used for semantic similarity matching to avoid wrong matching of relations and concepts.

3.8 CONCLUSION

Ontologies have become a powerful tool for text understanding. In this chapter a novel scheme for building ontologies from unstructured text is proposed based on considering semantic roles. Matching the semantic roles of concepts gives an additional feature for efficient merging of sub-ontologies leading to efficient construction of final ontology for better and more correct understanding of text. The rules required for various modules are designed and represented and the approach has been implemented using Java technology. The performance of the system is evaluated by looking coverage of concepts and relationships in the final ontology. The experiment shows that the ontology of the documents obtained by this scheme achieve F-score of 93.2 for concepts and 88.9 for relationships indicating good results.

CHAPTER 4
DESIGN OF ONTOLOGY ENRICHER

4.1 INTRODUCTION

Ontology gives annotations in rdf or rdfs formats which are used for providing intelligent services like information retrieval, question answering etc. These services can perform better if ontologies being used contain extra information about the concepts. This extra information gives an idea about the context of the concept and for this different levels of details can be added to ontology. But enriching the ontology is tedious and time-consuming task [37]

In this chapter, we propose a technique by which ontology can be extended with specific knowledge to provide more information about the constructs of ontology. Specifically, in this proposal, the ontology is enriched by providing the class labels for data properties extracted during the generation of ontology. For example in a sentence "The red car belongs to Zoe". The concept *car* has a data property *red*. This data property will be labelled with a tag *"colour"* which is the name of the class label we have defined for *red*.

4.2 BASIC APPROACH

We pursue the following steps to enrich ontology:

i) A dictionary of adjectives is formed and labels for the adjectives in the dictionary are gathered which organize these adjectives into different classes such as colour, condition, personality etc.
ii) We construct ontology for a text document by our proposed technique as given in Chapter 3 where the adjectives in sentences are extracted as data properties of the concepts.
iii) These data properties are mapped to the class labels from the dictionary of adjectives.

4.3 ARCHITECTURE OF ONTOLOGY ENRICHER

The architecture of the proposed ontology enricher depicted in Figure 4.1 has following components:

1. Ontology Generator
2. Mapping Tables
3. Data Property Classifier

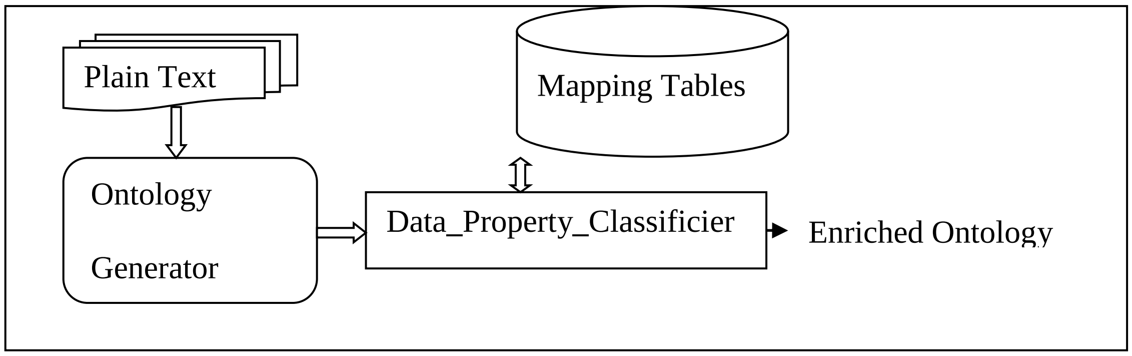


Figure 4.1: Architecture of Ontology Enricher

The description of each component is given as follows:

1. Ontology Generator

In this module, a basic ontology is generated by using our proposed approach as given in chapter 3 of this thesis.

2. Mapping Table

Following Categories are defined for extracted data properties in the mapping table:

i) Colour
ii) Taste
iii) Touch

iv) Relation

v) Quantity

vi) Personality

vii) Feel good

viii) Feel bad

ix) Condition

x) Appearance

xi) Numbers

xii) Size

xiii) Shape

xiv) Evaluation

xv) Sound

xvi) Nation

A total of 2360 adjectives belonging to these classes are extracted from dictionaries and stored in database in the tables named by these classes. A few data properties are shown in Table 4.1 along with their class and total number of data properties belonging to that class.

Table 4.1: Mapping Table

Data Property Class	**Data_Properties**	**Total Number**
Color	Red, maroon, cyan, Blond, Antique_Bronze, Celadon_Green, Tan, Canary_Yellow, Bondi_Blue, Blizzard_Blue, Brown_Sugar, Cadmium_Green, Dark_Byzantium, Blue, Magenta_Violet, Mikado_Yellow, Sinopia, Powder_Blue, Falu_Red, Sacramento_State_Green, Carmine, Heliotrope_ Magenta, Cerise, Roast_Coffee, Red-Brown, Pink-Orang, Cocoa_Brown, Palatinate_Purple, Red,Mystic_Maroon, Light_Moss_Green, Old_Moss_Green, Lapis_Lazuli,Nickel, Dark_Magenta etc.	1201
Quantity	Abundant, bountiful, cumbersome, Empty, just, enceinte, ending, extra, terminal, myriad, good, big, safe, break, closely, declamatory, prominent, tumid, large, turgid, Heavy, numerous, stopping point, full, penny-pinching, cheeseparing, confessedly, great, last, many, estimable, bountiful, a few, gravid, few, empty, lowest, bombastic,	32

	orotund, magnanimous, utmost, effective, a couple of, substantial, light, fill up, well etc.	
Personality	secluded, brave, truculent, life-threatening, mystic, abusive, Arcanum, private, loner, unbiased, bright, naughty, buck private, sober, unplayful, aggressive, determined, cowardly, selfish, renowned, presumptuous, clandestine, unavowed, witty, secret, ole-and-corner, zany, thrifty, ambitious, knowledgeable, mysterious, hidden, pleasing, hush-hush, sincere, mystical, disagreeable, , good, grievous, , confidential, hesitant, successful, fearless, punctual, combative, generous, evil, voracious, underground, mystery, warm, placid, surreptitious, jealous, instinctive, closed_book, grave, helpful, occult, , severe, wise, talented, diligent, dangerous, frank, gifted, privy, hugger-mugger, amused, cruel, individual, sedate, orphic, serious, harmonious, cloak-and-dagger, undercover, enigma etc.	44
Feel good	cheerful, fine, ager, obedient, friendly, good, safe, lucky, delighted, break, courageous, elated, fill_up, respectable, admittedly, true_up, trustful, delightful, beneficial, right, cooperative, nigh, dear, close, leery, stuffy, avowedly, expert, proficient, unspoiled, hilarious, honourable, net, full, penny-pinching, cheeseparing, confessedly, adept, trade_good, last, healthy, estimable, unspoilt, honest, glorious, fantastic, secure, well-disposed, unfeigned, encouraging, , thankful, snug, lively, lawful, sober, lastly, salutary, life-threatening, joyous, live, victorious, dependable, calm, agreeable, rightful , happy, wary, relieved, finis, charming, favourable, survive ,upright, true, sound, kind, utmost, reliable, excited, zealous, endure, , comfortable, confining, goodness, funny, genuine, faithful, skillful, soundly, ,truthful, playful, nice, energetic, enchanting, effective, enthusiastic, well, skilful, thoroughly etc.	46
Feel bad	envious, tired, confused, defeated, hungry, embarrassed, fierce, abashed, dizzy, frantic, homeless, helpless, obnoxious, frightened, arrogant, anxious, thoughtless, angry, defiant, abhorrent, nutty, itchy, bored, wicked, lazy, lonely, foolish, testy, disgusted, sore, uptight, repulsive, ashamed, awful, weary, condemned, scary, troubled, depressed, ill, terrible, worried, outrageous, bad, disturbed, jittery, grumpy, grieving, annoyed, panicky etc.	49
Condition	uninterested, real, crazy, tough, outstanding, abnormal, alive, doubtful, wandering, inquisitive, sometime, impossible, curious, careful, annoying, open, aberrant, expensive, previous, better, different, busy, erstwhile, abiding, authoritative, odd, concerned, crucial, one-time, tame, late, occupy, aberrational, other, worry, easy, frail, abeyant, brainy, concern, ablated, interested, difficult, poor, clever, innocent, puzzled, powerful, horrible, quondam etc.	54
Appearance	alluring, beautiful, fair, cute, alien, dynamic, fancy, bloody, cheerful, elegant, thoughtful, spotless, gleaming, long, blushing, graceful, smiling, ugly, perfect, unusual, zaftig, stormy, crowded, distinct, nervous, muddy, grotesque, cultured, jolly, confident, wonderful, extraneous, attractive, snobbish, gorgeous, poised, lovely, adorable, excited, timid, colorful, magnificent, alamode, homely, alert, pleasant, motionless, tense, vivacious, drab, dark, precious, hurt, filthy, dull, plucky, handsome, gentle, misty, glamorous, etc.	79

number	one, two, three, four…fifty three.. , hundred, thousand, million, billion etc.	
Size	massive, initiative, tiny, lowly, thick, minuscule, small-scale, low_gear, huge, little, first_gear, gravid, number_one, pocket-sized, inaugural, long, belittled, immense, showtime, low, mammoth, scrawny, minor, thin, bombastic, maiden, petite, big, fat, small, modest, tall, magnanimous, offset, declamatory, orotund, foremost etc..	**21**
Shape	cylindrical, oval, first_base, narrow, three-dimensional, helix, flat, crested, anguilliform, serriform, pisiform, soliform, minuscule, two-dimensional, pisciform, ,patelliform, acinaciform, cymbiform, wraparound, pyriform, caudiform, muriform, tubiform,small, round, forked, calcariform, crescent, etc.	**537**
Evaluation	yearly, up-to-date, old, modern, one-year, brief, raw, sometime, one-time, yearbook, weekly, slow, new fangled, erstwhile, monthly, young, fast, late, brand-new, ancient, freshly, quarterly, new, newly, previous, abbreviated, quondam, onetime, rapid, second-hand, early, recent_epoch, swift, former, recent, fortnightly, novel, annual, unexampled, annually, quick, fresh, latest, other, etc.	**30**
Sound	mute, dumb, unsounded, hissing, harsh, voiceless, hushed, faint, silent, calm, thundering, noisy, shrill, cooing, whispering, melodic, squealing, blaring, squeaking, tacit, mum, purring, husky, resonant, sonorous, soundless, melancholic, understood, soft, raspy, quiet, screeching, deafening, moaning, loud etc.	**31**
Nationality	Malagasy, Estonian, North Korean, South Korean, Chadian, Grenadian, Palauan, Kyrgyz, Kenyan, Belgian, New Zealander, Surinamer, Bulgarianm swiss_people, Indian, Pakistani, Herzegovinianm, Turkish, French, Fijian, Liberian, Armenian, SaoTomean, Eritrean, Qatari, Haitian, Mauritanian, East Timorese, Finnish, Canadian, Kittianand, Nevisian, Argentinean, Albanian, Omani, Comoran, Lebanese, Maltese, North African, Bahraini, Serbian, Middle Eastern, Emirian, Italian, Malian, Ivorian, Icelander, Panamanian, Zambian, Moroccan, Guinea-Bissauan, Burundian, Tajik, Dutchman, Scottish, Senegalese, Austrian, Beninese, Tongan, Gabonese, Marshallese, Kuwaiti, Guinean, Slovenian, Rwandan, Azerbaijani, Bruneian, Nauruan, San Marinese, Bahamian, Indonesian, Peruvian, Leonean, Burkinabe, Norwegian, Slovakian, Monacan, Malawian, Ugandan, Thai, European, Ghanaian, Kazakhstani etc.	**236**

3. Data Property Classifier

This component takes ontology as input and works on data properties of the ontology. Naı̃ve string matching is used to match each extracted data property with some adjective stored tables in the database. If a match is found, that data property is related to the name of the table in which the match is found. The label of this relation is named as '*has class*'. The algorithm for the same is shown in Figure 4.2 as follows:

```
Algorithm data_property_classifier()
Input: sentence tokens list with dependency tags next to each token in the list,
adjective database, mapping table
Output: concepts properties with class labels
Begin
1. for each token in the sentence list
  2. for each concept  from concept_ list
     a) analyze the dependency tag "amod" for extracting properties of concepts
     b)extract the concept property
 3. look for a match of  the extracted data_property in the adjective_database
 4. If match found, extract the name of table in which match is found
5. store the data property along with the name of table with label "has_class"
  end for
end for
```

Figure 4.2: Algorithm for Data Property Classifier

The algorithm takes as input the sentence tokens list with dependency tags next to each token in the list, adjective database and mapping table we defined. It begins by processing each token of the sentence and the concepts extracted already. In step 2 data property is extracted by analysing the dependency tag "amod". The extracted data property is checked for its occurrence in the adjective database. If a match is found in step 4, the data property is mapped to the name of the table where the property was found and stored with a label "has class" in step 5.

4.4 WORKING EXAMPLE OF ONTOLOGY ENRICHER

For a text document given as:

Pakistani singer Ghulam will perform at famous musical festival that begins at Sankatmochan temple.

We will construct an ontology for this text document using the technique proposed by us in Chapter 3 as shown in Figure 4.3. We will extract concepts along with their semantic roles and data properties of these concepts and object properties.

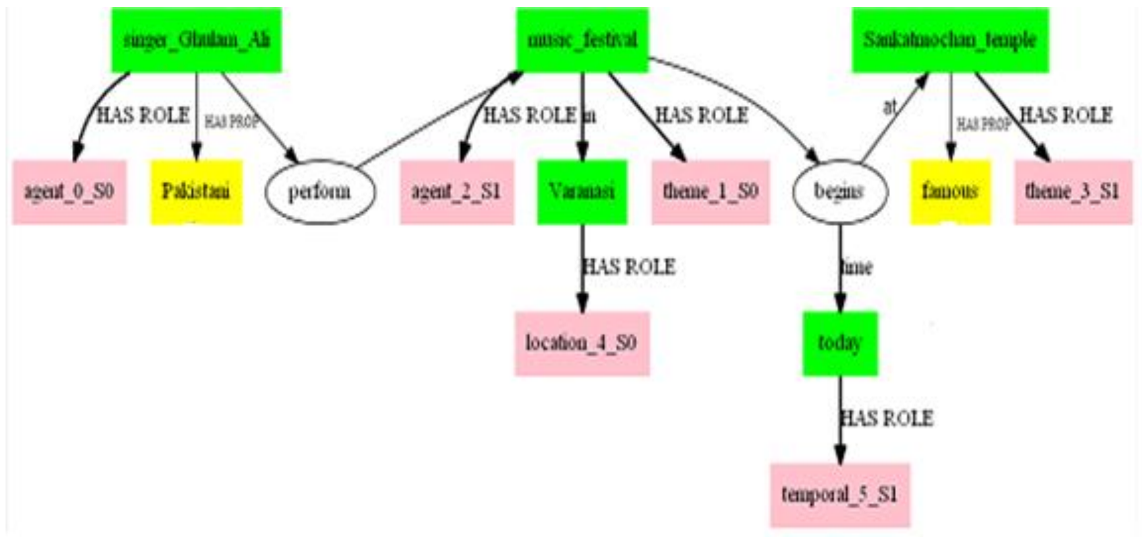


Figure 4.3: Ontology of Text Document

Here, extracted data properties will be Pakistani and famous. These data properties are given as input to the data property classifier that process it using the mapping table and allots their respective classes which are Nation and Condition respectively. We get the following output as shown in Figure 4.4.

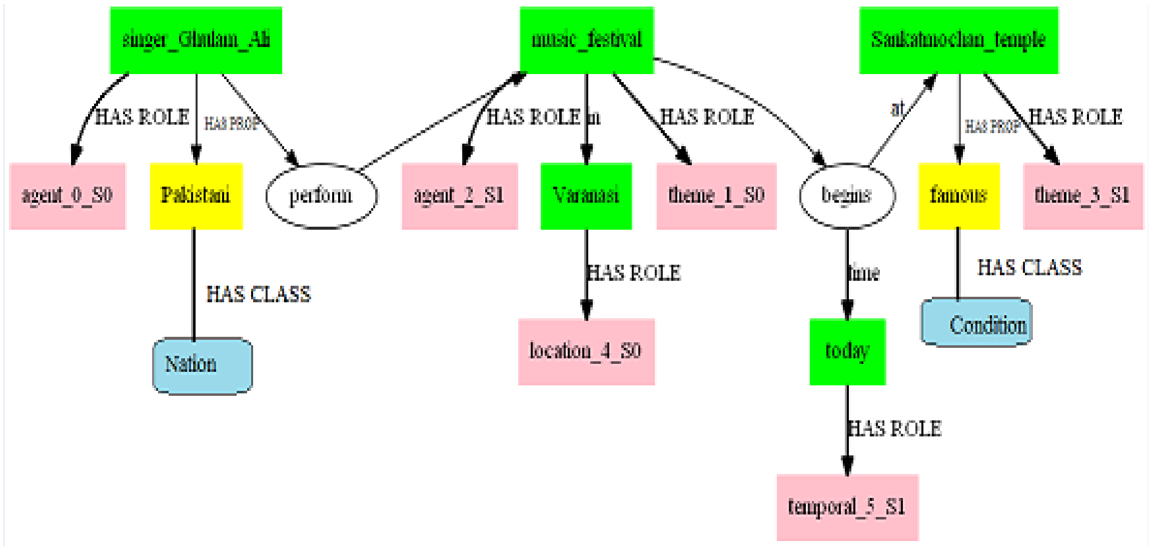


Figure 4.4 : Enriched Ontology of Text Document

4.5 IMPLEMENTATION DETAILS

We have implemented this work using NetBeans 8.0.2 which is using Apache Derby as relational database which is bundled with NetBeans. Apache Derby is based on Java, JDBC and SQL standards.

4.6 CONCLUSION

We propose a technique in this chapter that extends and enriches ontology with explicit knowledge so that the intended meaning of the concepts is more expressive. This process is performed during the course of constructing the ontology in which the ontology is enriched by providing the class labels for data properties extracted.

CHAPTER 5

AUTOMATIC DOCUMENT SUMMARIZATION USING ONTOLOGY

5.1 INTRODUCTION

Summarization of text is a necessity as there is a large amount of data on the web expressing the same ideas. It requires deciding which sentences or phrases are to be chosen such that they show the main ideas in the document. Summarizing large texts manually is both costly and time consuming. Automatic text summarization is a process of making a coherent summary that retains the most important points of original document using a computer program. It is a method for data reduction which enables users to reduce the amount of text that must be read to gather the essential information [53].

In previous contributions Ramanujam and Kaliappan [52] use the Naïve Bayesian Classification and the timestamp concept, [53] have presented a technique using unsupervised deep learning approach using Restricted Boltzmann Machine, [51] use latent semantic analysis and fuzzy logic system. TextRank [66] creates a graph where nodes represent sentences and edges are added between nodes and they specify the similarity value between the two nodes (sentences) it connects. Top ranked sentences are then chosen to form the summary. Some researchers have used ontology in the process of summarization such as [46] use a hierarchical ontology to generate summaries. Ontological knowledge is used by [47] also to generate document summary. [48] depend upon YAGO ontology to evaluate and select sentences from documents. [45] Ragunath and Sivaranjani present an idea for ontology-based summarization to compute a set of features for each sentence based on the output of the hierarchical classifier.

5.2 PROBLEMS IN EXISTING APPROACHES

The work proposed in this chapter attempts to resolve the following problems which have been found in previously done work by other researchers, as is also mentioned in the Chapter 2.

5.2.1 Overlapped Information

In most of the contributions focusing on extractive summarization where whole sentences are included in summary may lead to overlapped information in summary. As semantic structure of sentence and semantic relationships between sentences is not taken into account, these methods may not be able to identify sentences which are semantically equivalent. Thus, the final summary would contain redundant information. [67] [68]

5.2.2 Dangling Co-references

Moreover there may be problem of ‘dangling co-references’ as sentences containing pronouns may lose their relevance if extracted out of context. [69]

5.2.3 Ignoring The Semantic Structure Of Sentence

Aforementioned methods also treat sentences as bag of words and are unable to understand text deeply. Ontology based approaches are also unable to capture the full semantic structure of the sentence as they use only hierarchical ontologies as discussed by Hennig, Umbrath & Wetzker [46] or hierarchical classifiers for mappings as used by Ragunath and Sivaranjani [45] ignoring the non-hierarchical ontologies.

5.3 BASIC APPROACH

The goal of summarization is to achieve high similarity of the summary information to the original document and lesser redundancy. Two major categories of text summarization are (i) extractive and (ii) abstractive summarization [67]. Extractive summarization techniques select important sentences from the text to be extracted for generating summary. Importance of sentences is calculated on the basis of some features such as position of the sentence in the document, term frequency, lexical chains etc. For contents such as news articles and reviews about a product, there is a lot of redundancy. Using extractive summarization for this kind of content may not be a good idea as extractive summaries may contain unnecessary information. For this kind of content abstractive summaries provide a concise and compact idea of the content. Abstractive summarization is able to generate sentences other than the original sentences in the given text. These new sentences have to be grammatically correct and able to convey the summarized information in a consistent way. This technique requires deeper text understanding to build some representation of text before generating the summary.

The approach proposed by us combines both methods to generate summary by first extracting features of the text documents to obtain an extractive summary and then creating ontology for the extractive summary document and rephrasing the sentences using this ontology. The proposed technique is novel because this technique gets success in removing the redundant information from the extractive summary of the text giving semantically correct yet more concise information.

Specifically, our approach of summarizing a document is a hybrid technique that involves few sub-steps:

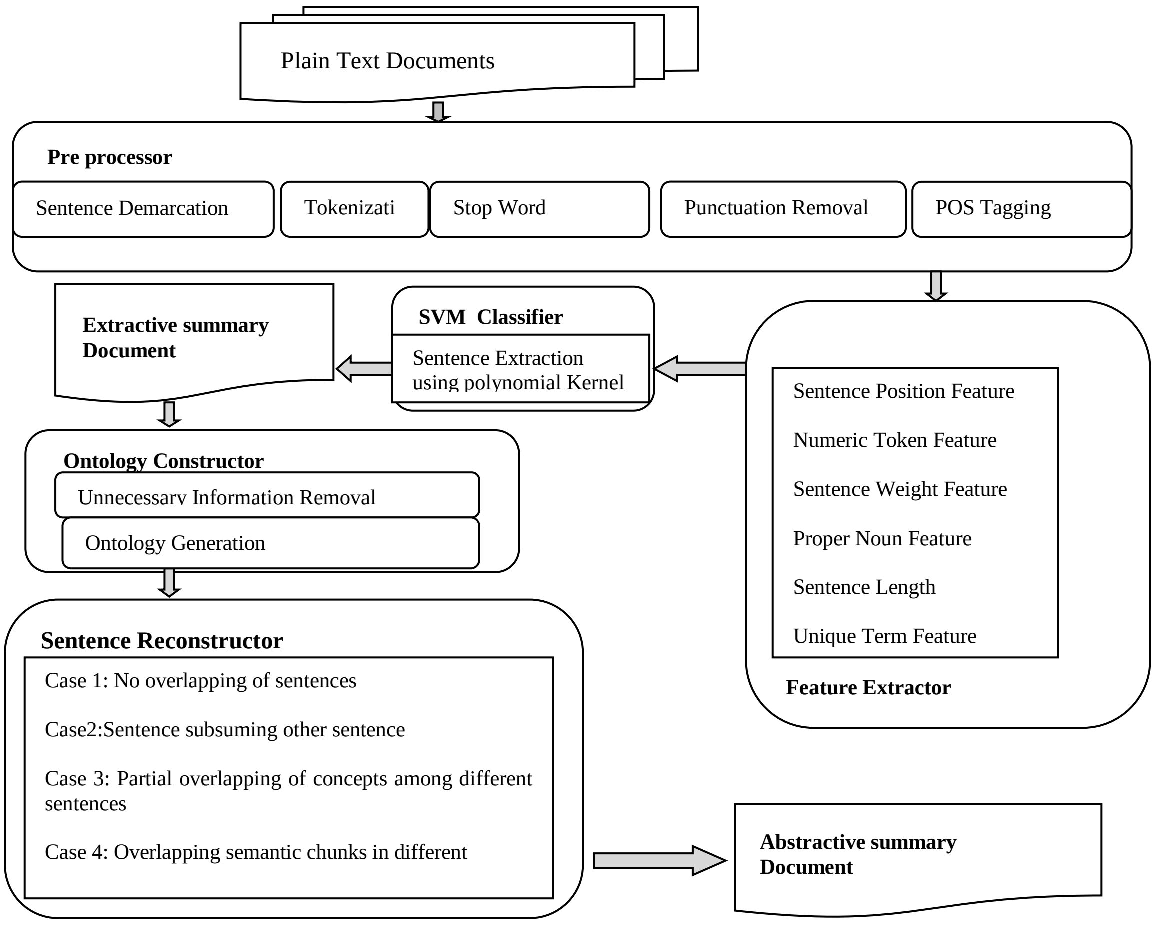
i) Extracting some statistical features from the text and using SVM classifier to generate extractive summary.

ii) Generating text document ontology for this extractive summary keeping into account the hierarchical and non-hierarchical relationships among the constituents of sentences in the extractive summary document. This is done by identifying and merging semantically similar sentences and concepts.

iii) Afterwards the sentences are reworded or reconstructed from the ontology to attain an abstractive summary.

5.4 ARCHITECTURE OF THE PROPOSED SYSTEM

The architecture of the proposed system to generate the abstractive summary as shown in Figure 5.1.



Plain Text Documents
Pre processor
Sentence Demarcation
Tokenizati
Stop Word
Punctuation Removal
POS Tagging
Extractive summary Document
SVM Classifier
Sentence Extraction using polynomial Kernel
Sentence Position Feature
Numeric Token Feature
Sentence Weight Feature
Proper Noun Feature
Sentence Length
Unique Term Feature
Feature Extractor
Ontology Constructor
Unnecessarv Information Removal
Ontology Generation
Sentence Reconstructor
Case 1: No overlapping of sentences
Case2:Sentence subsuming other sentence
Case 3: Partial overlapping of concepts among different sentences
Case 4: Overlapping semantic chunks in different
Abstractive summary Document

Figure 5.1: Architecture of the Proposed Text Summarizer

1. Pre-proccesor

The text document is prepared to be processed for summarization by performing some pre-processing of the document such as tokenization, stop words removal, punctuation removal etc.

2. Feature Extractor

Feature extraction is the process of transforming the input data (sentences) into a set of features [70] . This is done here to perform the desired task of summarization using the reduced representation instead of the full size input. The preprocessed text document is processed further by Feature extractor so that some sentence specific features can be extracted from the text documents which are given as input to the next module SVM classifier. These features are: Sentence Position Feature, Numeric Token Feature, Sentence Weight Feature and Proper Noun Feature.

3. SVM classifier

The SVM classifier extracts the sentences to be included in summary. The feature set extracted in the previous module for the document is used to train the classifier and then extract the important sentences from the document and thus generate the extractive summary.

4. Ontology Constructor

The extractive summary generated by SVM classifier may contain overlapped information such as different sentences but containing semantically similar information. This extractive summary document is further processed to construct an ontology. For constructing ontology from that extractive summary document we identify the concepts in the document, their properties and relations among concepts. Some additional information i.e. semantic roles these concepts are playing in each sentence of the text is also attached while constructing ontology. For this, the sentences of extractive summery document are parsed using Stanford dependency parser that provides dependency tags attached with each term. By utilizing these dependency tags new tags are formed which specify the concepts, relations, semantic roles and properties. The concepts and relations are represented in an ontological structure with additional information such as semantic roles and properties of concepts in the sentence.

5. Sentence Reconstructor

The ontology thus built in previous module is used to reconstruct the sentences to generate an abstractive summary of the text document in Sentence Reconstructor module. As the ontology of the extractive summary document removes the un-necessary and redundant phrases or sentences, the sentences reframed by this module constitute the concise representation of the text document. The Sentence reconstruction process may also use additional words such as *"then"* and *"and"* to join two sentences or parts of sentences. Inverse co-reference resolution is also performed in this module so that interpretation of the sentences in summary is in consistency with their interpretation in the original text.

5.5 DETAILED DESIGN OF SYSTEM

This subsection describes the methodology of components designed for generating the summary along with the type of data and the data structures each component uses and also gives the flow and process of generating the ontology by providing the algorithms designed for modules of the system. The detail of each module of the system is as follows:

5.5.1 Pre-processor

The text to be summarized has to go through some pre processing steps so that this text can be used for extracting features given as follows:

i) Sentence demarcation

ii) Tokenization

iii) Stop Word Removal

iv) Punctuation Removal

v) Part Of Speech Tagging

A brief explanation of each step is given as follows:

i) Sentence demarcation

In this step the complete text is divided into sentences using NLTK python library.

ii) Tokenization

This step tokenizes the sentences to generate tokens using NLTK python library. These tokens are used to detect keywords and key phrases in the text.

iii) Stop Word Removal

Stop word are highly frequent words that do not carry any information. These are filtered out before processing the text. We have filtered out stop words list from the NLTK corpus.

iv) Punctuation Removal

Punctuation marks are also removed to be not included in text features count.

v) Part Of Speech Tagging

In part of speech tagging according to the category of words i.e. noun, verb, adjective etc. words are tagged. We have used Stanford Parser to perform the part of speech tagging.

- **Data Design for pre-processor**

The pre-processed text is stored in primary memory for further processing by next modules in the system.

- **Algorithm Design for Pre-processor**

The algorithm for pre-processing the text document to be summarized is given in Figure 5.2.

```
Algorithm sum_pre_processor()
Input:  text document, NLTK python library
Output: pre-processed text document
Begin
        for each sentence of the text document
        1.  demarcate sentence
        2.  tokenize
        3.  remove stop words
        4.  remove punctuation
        5.  perform part of speech tagging to each token
         end for
End
```

Figure 5.2: Algorithm for Sum_pre_processor

The algorithm takes the text document as input and uses NLTK python library for its processing. In step 1, the sentence is marked for its beginning and ending. Step 2 tokenizes the sentence. Stop words and punctuation marks are removed from the sentence in step 3 and 4 respectively. Each token identified in step 2 of the algorithm is provided part of speech tag in step 5. The output of the algorithm is the pre-processed document which is given to the Feature Extractor module.

5.5.2 Feature Extractor

The pre-processed text is given to the next module which is Feature vector calculator. Feature vectors are generated for each sentence in the pre-processed text document. The value of each feature lies between 0 and 1. These feature vectors are used to form the feature matrix. Following features are extracted from the text. [53]

i) Sentence Position Feature

The position of the sentence in the document determines its relevance. As the first sentence of text document is supposed to contain important information, it is given a score 1. Also last sentence of the sentence is the concluding sentence, it is also given score 1.

Sentence_Pos = 1, for the first or last sentence of text document.

Sentence_Pos = cos((Sentence_Pos -min)((1/max)-min)),* for the rest of the sentences.

Where, *Sentence_Pos* is the position of sentence in the given text.

min is calculated as (threshold*N)

max is calculated as (threshold*2*N)

N is total number of sentences in document.

threshold is calculated as (0.2*N)

ii) Numeral Token Feature

This feature is calculated for the sentences containing numeral tokens by dividing total number of numeral tokens in that sentence with total number of words in that sentence.

$$\text{Numeral\_token\_feature}^{S}i = \textit{num\_numeral}^{S}i\ /\textit{Slength} \qquad \text{Equation 5.1}$$

where, *num_numeral*^{S}*i* is number of numeric tokens in ***i**th* sentence.

Slength is total number of words in sentence

iii) Weight of the sentence

This feature is calculated by summing the frequencies of all words in sentence and dividing it by the length of sentence.

Freq_word = Freq_word/Maximum Frequency Value Equation 5.2

where, *Freq_word* is the frequency of a word occurring in a sentence.

Maximum Frequency Value is the frequency of the word occurring the maximum times in the document

Sentence_Weight= Sum(Freq_word of all words in the sentence)/length of sentence

Equation 5.3

iv) Proper Noun Feature

Proper nouns refer to the named entity. Part-of speech tagging of each sentence performed by pre-processor help determine proper nouns. This feature gives significance to those sentences, which contain more proper nouns.

v) Sentence Length Feature

Sentence length can help to determine the amount of content in the sentence.

Sentence_length = Number of words/len_max

Where, *len_max* = Length of biggest sentence in the document

vi) Unique Term Feature

This feature is used to calculate the number of unique terms in a sentence. It gives weightage to those sentences that contain new information.

Unique_Term = number of unique term in that sentence/Length of the sentence

- **Data Design for Feature Extractor**

Output Data Format

The output of this module will be a set of feature sets. The format of a feature set corresponding to a sentence will contains numeric values calculated as discussed. For example, if we take following sentence which is first sentence of the text document taken in section 5.6:

The highest flood peaks on the Xijiang and Beijiang Rivers have passed, said Zhu Senlin, governor of south China's Guangdong Province.

The feature set for this sentence will be calculated by this module is as follows:

X=[1.0, 0.0, 0.32, 0.5102040816326531, 0.1366666666666667, 0.24]

Here, sentence position feature=1, for being the first sentence of text.

Numeric token Feature = *num_numeric /length*
= 0/25=0

Proper noun Feature = number of terms tagged as nouns/length
= 8/25=0 .32

Sentence length Feature = length of sentence/length of the longest sentence
=25/49=0.5102040816326531

Weight of the sentence =sum *(Freq_word of all words in the sentence)/length of sentence*
= 3.41/25 = 0.1366666666666667

Unique term Feature= number of unique terms/length= 6/25= 0.24

For each document having N sentences in total, feature data will be having N feature sets. This feature sets are stored in a text file in secondary memory.

- **Algorithm Design for Feature Extractor**

The algorithm for extracting features as shown in Figure 5.3 begins with processing each sentence in the text document to extract the features.

```
Algorithm Feature_Extractor
Input sentences from text files
Output feature vector matrix
Begin
for each sentence in text files
Step1: //compute the position of sentence
        $SenPos = \cos\{(sentencePosition - minTh) * \left(\frac{1}{maxTh} - minTh\right)\}$
Step 2: //check for numeric enteries
        $Num\_val = \frac{Number\ of\ numerical\ terms}{length\ of\ the\ sentence}$
Step 3: //compute frequencies of each word
        $Freq_{term} = \frac{Frequency\ of\ the\ term\ in\ the\ document}{Highest\ term\ frequency}$
Step 4: // look for proper noun tags
        $Noun_{term} = \frac{Numbr\ of\ terms\ tagged\ as\ nouns}{length\ of\ the\ sentence}$
Step 5:// calculate the length of each sentence
        $Sen_{len} = \frac{Number\ of\ terms\ in\ the\ sentence}{length\ of\ longest\ sentence}$
Step 6://find the unique terms
        $New_{term} = \frac{Number\ of\ unique\ term\ in\ that\ sentence}{length\ of\ the\ sentence}$
  Step 7: //feature vector matrix preparation
   $Feature\_matrix[i] = [Senpos[i], Num\_val[i], Freq_{term}[i], Noun_{term}[i],$
$Sen_{len}[i], New_{term}[i]]$
 end for
End
```

Figure 5.3: Algorithm for Feature Extractor

Step 1 extracts the position of the sentence. Step 2 checks the sentence for any numeric value and normalizes its value by dividing it to the length of the sentence. Step 3 computes the frequency of each word of the sentence in the document and normalizes this value by dividing it to the highest term frequency. Step 4 checks for the number of terms which are proper nouns. This value is normalized by dividing it to the length of the sentence. Step 5 calculates the length of the sentence and normalizes its value by dividing it to the longest sentence in the document.. Step 6 finds the terms which are having frequency no more than one in the document. Step 7 generates the feature matrix using all above calculated values of features. These feature vector matrices are given to SVM classifier so that the extractive summary can be obtained as shown in the following sub-section.

5.5.3 SVM Classifier

As discussed in section of Chapter 2, Support Vector Machines given by Vapnik, V. [58] are supervised learning algorithms which show good performance for text categorization tasks [71] and can also be used for text summarization [72] [73] [74] [50]. We have used SVM classifier in our work to extract the important sentences from the text. We train the SVM to classify the sentences from the text document for summarization. The process of generating the extractive summary using SVM is done in following steps.

i) Training the SVM classifier

ii) Sentence Extraction

These steps are explained in detail as follows:

i) Training the SVM classifier

We take some text documents as training documents from DUC2002 dataset [75] which are pre-processed initially and the feature extraction for each sentence of the documents is performed generating the feature vectors of each sentence. Then we label feature vectors as positive or negative. Positive label is given to a feature vector if the sentence corresponding to that vector is present in the reference summary of that text document. Otherwise the label given to that feature vector is negative label.

We have used 500 random positive and negative example sentences from our dataset to train our SVM classifier. This trained classifier is used for extracting the important sentences for summarization.

ii) Sentence Extraction using Polynomial Kernel

The test document is then fed to this trained SVM classifier as input. For this non-linear decision surface of sentence extraction, kernel trick is applied to maximum-margin hyperplane and the dot product is replaced by the kernel function.

$$\mathbf{k}(\vec{\mathbf{x}}_{\mathbf{i}}\,,\mathbf{y}) = (\vec{\mathbf{x}}_{\mathbf{i}}\mathbf{y} + \mathbf{1})^{\mathbf{2}} \qquad \text{Equation 5.4}$$

This polynomial kernel have been very effective when applied to several tasks of natural language processing of a second degree with a value of C as 0.0001 [53]. We have used the same for extracting summary from text. The documents classified as positive are ranked according to their distance from the maximum margin hyper-plane. They will be assigned Y=+1 if the sentence and the output are the sentences corresponding to the top 35% sentences in the rankings.

This extractive summary is further shortened by constructing ontology of this extractive summary using the proposed system for generating ontology and reconstructing the sentences using this ontology to generate abstractive summary as explained in the next sub-section.

vi) Algorithm Design for SVM Classifier

The algorithm as shown in Figure 5.4 begins with generating the training data file by labeling the 500 sentences as positive by providing them label +1 or negative by providing them label -1 in Step 1. This is training file is used to train to the SVM classifier in Step 2.

```
Algorithm svm_classifier()
Input: feature vectors of Trainfile , ManualSumFile, test text file
Output: extractive summary of test text documents
Begin
Step1:  //generate training text file  train-500.txt
        for each sentence S having feature vector x[i] of TrainFile in
      ManualSumFile
        y=+1,corresponding to feature vector x[i]
        else
        y= -1,corresponding to feature vector x[i]

Step 2: // train the svm classifier
        X_train, y_train = load_svmlight_file("train-500.txt")
Step 3: load test text file
Step 4: Call  svm.SVC(kernel='poly', C=0.0001, degree=2, probability=False)
Step 5: Rank sentences according to their distance from hyperplane
Step 6: Take top one third of ranked sentences
End
```

Figure 5.4: Algorithm for SVM Classifier

The test text file whose summary is to be generated is loaded in step 3. In the next step SVM polynomial kernel of degree 2 is applied with value of C=0.0001. Step 5 ranks the sentences according to their distance from maximum margin hyperplane. We pick the top 7 sentences as extractive summary in step 6.

To remove redundancy among sentences in extractive summary we construct an ontology for the same by Document Ontology Constructor.

5.5.4 Document Ontology Constructor

The extractive summary is given to ontology generation module which further reduces the text by removing the transition words and unnecessary information during preprocessing. Ontology is generated for this reduced text by constructing sub ontologies for each sentence and then by merging these sub ontologies by using the rules proposed in ontology mapping and merging module in section 3.4. Here, relationships among sentences are described by the semantically similar concepts and object properties that are common in different sentences.

This is shown in the ontology using GraphViz [57] in Figure 5.6 having multiple edges among semantic chunks from different sentences or having same concept but different or same roles in multiple sentences. By generating the ontology from extractive text summary, we further merged the semantically similar sentences (fully or in some measure). This information is utilized in reconstructing the sentences by Sentence Reconstructor.

- **Data Design for Ontology Constructor**

The ontology constructed for the text of extractive summary will be stored in rdf format in the ontology repository.

- **Algorithm Design for Ontology Constructor**

The algorithm used for constructing ontology is same as given in section of Chapter 3. The only difference is the input text file which is extractive text summary document.

5.5.5 Sentence Reconstructor

The ontology thus constructed after removing unnecessary words or phrases or parts of sentences gives further concise representation of the extractive summary. In this ontology, similar sentences or parts of sentences are merged by finding the semantic and syntactic similarity among different sentences based on hierarchy and roles of the concepts in the sentences. This ontology is used now to reframe the sentences to obtain the abstractive summary. There can be following cases while rewording the sentences:

Case 1: No overlapping of sentences

Sentences are taken as such in the summary.

Case 2: Sentence subsuming other sentence

The longer sentence is taken into summary and the subsumed sentence is discarded.

Case 3: Partial overlapping of concepts among different sentences

This can be the case where a concept is having same role in different sentences but is part of different semantic chunks (concept-verb-concept). Connector such as *and* or *then* is used.

Case 4: Overlapping semantic chunks in different sentences

A semantic chunk can be overlapping in two sentences and can be used as a connector for merging two sentences.

The abstractive summary thus obtained is further processed to perform inverse co reference resolution as the final step of sentence reconstruction to get the final abstractive summary.

Inverse Co reference Resolution

If a merged sentence contains entity name more than once in same or consecutive sentences, the later one is converted to the relevant pronoun. This process is inverse co reference resolution. The output of this module is the final coherent abstractive summary also resolving the problem of “Dangling Co-references”.

- **Algorithm Design for Sentence Reconstructor**

The algorithm designed for reconstructing or reframing the sentences shows the process of constructing abstractive summary from the extractive summary and the ontology of that extractive summary as shown in Figure 5.5.

```
Algorithm sentence_reconstructor()
Input  object property table of each sentence of extractive summary, object property
           table of final ontology , extractive summary
Output reframed sentences for abstractive summarization
Begin
1:  for each sentence Si of object property table
        match each row to object property table of final ontology
2:    if all rows are matched                         //case 1:subsumption
            discard the sentence Si from summary
3:    if no rows are matched                      // case 2: no overlapping
            keep the sentence in summary
4:    if some rows have  matching concept –verb-concept triplets //case 3
            merge the sentences by keeping overlapped concept-verb- concept triplet
5:    if concepts are matching in but relations and semantic roles are different
            merge the sentences by keeping that sentence and using connector "and"
        with the  other matched sentence                                  //case 4
 6 : perform inverse co reference resolution
End
```

Figure 5.5: Algorithm for Sentence Constructor

This algorithm takes the extractive summary of the text document, the object property tables of each sentence of the document and the object property table of the final ontology of the document. The algorithm begins by processing the object property table of each sentence as the step 1. Step 2 ensures if all rows of object property table of a sentence are matching with those of final object property table then this sentence is being subsumed by another longer sentence and is not taken in final summary. Step 3 checks If there is no overlapping of sentences then the sentence is kept as such in the summary. Step 4 finds whether there is overlapping among sentences. Step 5 merges the parts of sentences which are matching. Step 6 merges the sentences by using the connector word "and". Step 7 performs the inverse co reference resolution to convert repeated entity names in consecutive sentences to suitable pronoun.

5.6 IMPLEMENTATION EXAMPLE

We have implemented our proposed system using python. We trained our SVM classifier over DUC2002 [75] dataset for summarization. For a text document shown below, pre-processing is performed on it in which stop words and punctuation marks are removed after tokenization of the words of sentences and these words are tagged with part-of-speech tags. This preprocessed text is processed further for extracting features as described in section 5.5.2.

Input Text Document

The highest flood peaks on the Xijiang and Beijiang Rivers have passed, said Zhu Senlin, governor of south China's Guangdong Province. While inspecting flood control and relief work in Qingyuan city on the Beijiang River yesterday, he attributed Guangdong's success in combating this flood -- almost the biggest in 100 years -- to concerted efforts by the armymen stationed in Guangdong and local residents. More than 200 people lost their lives in the natural disaster, which destroyed 189,000 rooms and ruined crops on 1.2 million hectares. The flood was caused by successive torrential rainstorms in the Xijiang and Beijiang River valleys in early and middle June. Major flood monitoring stations on the two rivers recorded their highest water levels, all four meters above the danger mark. Local governments at various levels in the province have paid close attention to flood control work, and leading government and communist party officials of different localities have gone to the flood-fighting front. No breaches of major embankments or reservoirs were reported despite the most serious flood in a hundred years, effectively protecting the safety of the provincial capital, Guangzhou, and the Pearl (Zhujiang) River Delta. But the losses caused by the flood were quite serious, said the governor. According to him, 11 million people in the province's nine cities and 55 counties were affected, and more than 200 people died in the natural disaster, with 189,000 rooms destroyed and 1.2 million hectares of crops ruined. The direct economic losses were set at 10.2 billion yuan. The governor warned that though the flood danger *had receded, the determination to fight possible further floods could not slacken, as this is just the beginning: the main flood season, which usually begins in late July and early August, has not yet arrived. He urged local officials to be on constant alert against further possible floods and be meticulous about flood prevention and control measures, while doing their utmost to help flood victims, assisting them to resume production as soon as possible and maintaining social stability.*

This feature set obtained from Feature Extractor is given as input to the trained SVM classifier which ranks these sentences according to their distance from the hyper-plane. The farther the sentence from hyper-plane, more confidently the classifier predicts it a positive example.

From the sentences which were labeled positive we have taken top 35% of ranked sentences as extractive summary as shown here.

Extractive Text Document

But the losses caused by the flood were quite serious, said the governor. The flood was caused by successive torrential rainstorms in the Xijiang and Beijiang River valleys in early and middle June. More than 200 people lost their lives in the natural disaster, which destroyed 189,000 rooms and ruined crops on 1.2 million hectares. Local governments at various levels in the province have paid close attention to flood control work, and leading government and communist party officials of different localities have gone to the flood-fighting front. According to the governor, 11 million people in the province's nine cities and 55 counties were affected, and more than 200 people died in the natural disaster, with 189,000 rooms destroyed and 1.2 million hectares of crops ruined. The direct economic losses were set at 10.2 billion yuan. The governor warned that though the flood danger had receded, the determination to fight possible further floods could not slacken, as this is just the beginning: the main flood season, which usually begins in late July and early August, has not yet arrived.

This extractive summary document is converted into document ontology using technique discussed in the chapter 3 and due to space constraints, a small part of that ontology is shown using GraphViz tool in Figure 5.6.

After applying the cases on the obtained ontology, as discussed in sentence rewording section we get the Final summary as shown below.

Final Summary

The flood was caused by successive torrential rainstorms in the Xijiang and Beijiang River valleys in early and middle June. Local governments at various levels in the province have paid close attention to flood control work, and leading government and communist party officials of different localities have gone to the flood-fighting front. 11 million people in the province's nine cities and 55 counties were affected, and more than 200 people died in the natural disaster, with 189,000 rooms destroyed and 1.2 million hectares of crops ruined. The direct economic losses were set at 10.2 billion yuan. The governor said the losses caused by the flood were quite serious and warned that though the flood danger had receded, the determination to fight possible further floods could not slacken as this is just the beginning the main flood season which usually begins in late July and early August has not yet arrived.

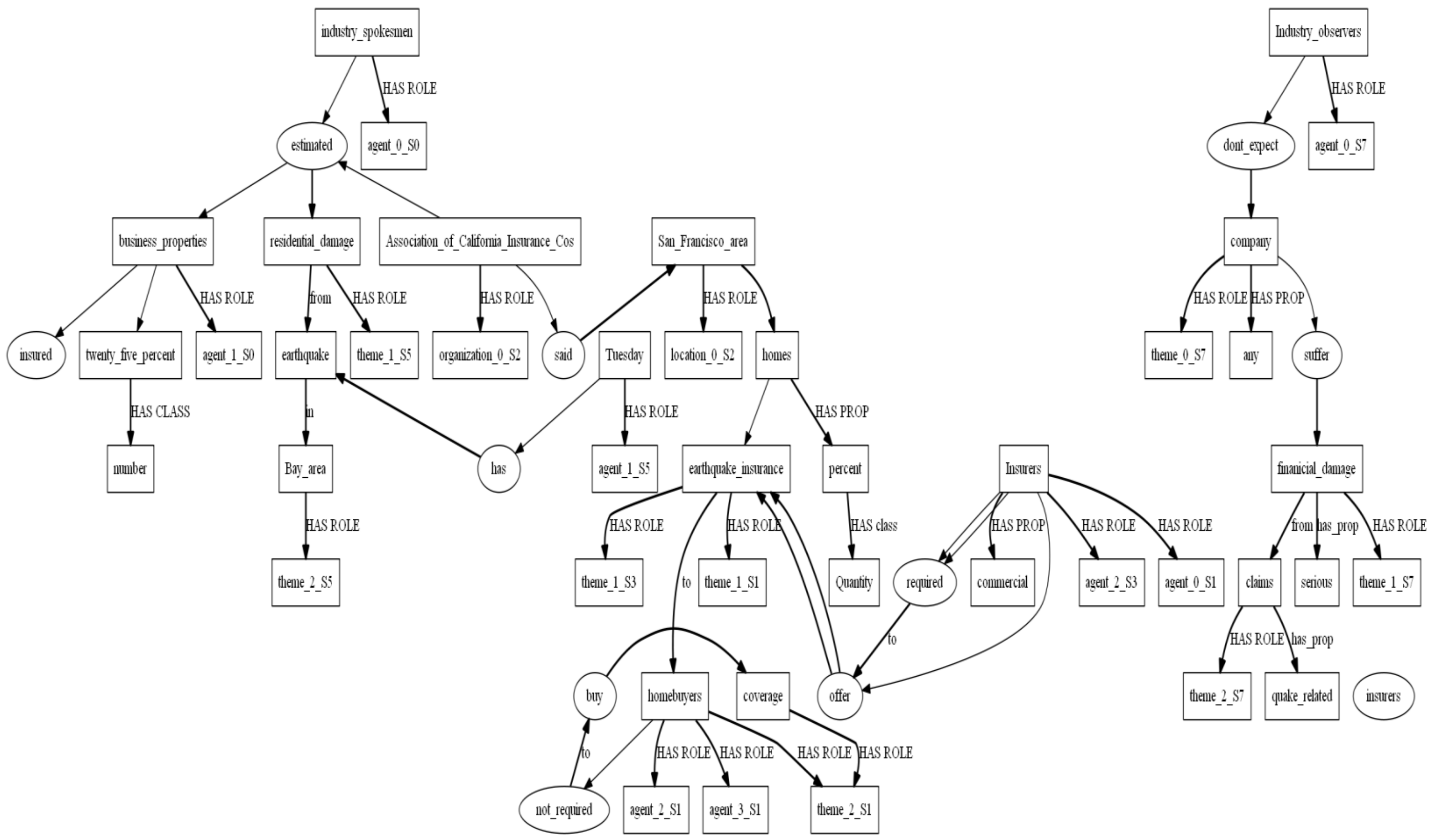


Figure 5.6: Extractive Text Document Ontology

Once our system has generated the summary, we want to know if this resembles the manually created summary for the same document. The manual (reference) summary as given in the DUC2002 dataset for the given document is shown as follows:

Manual/Reference Summary

The highest flood peaks on the Xijiang and Beijiang Rivers have passed, said Zhu Senlin, governor of south China's Guangdong Province. No breaches of major embankments or reservoirs were reported despite the most serious flood in a hundred years. Eleven million people in the province's nine cities and 55 counties were affected, more than 200 people died, 189,000 rooms were destroyed and 1.2 million hectares of crops were ruined in this natural disaster. Economic losses were set at 10.2 billion yuan. The really bad news is that this is just the beginning. The main flood season has not yet started.

We can analyze here that our system generated summary contains most of the sentences that are present in the manual summary.

5.7 EXPERIMENTS AND PERFORMANCE ANALYSIS

Evaluating an automatically generated summary is quite difficult task. We have used the ROUGE method (the measure adopted by DUC [76] as the standard for assessing the summary coverage) that calculates the intersection of n-gram, word pairs and word sequences between candidate summaries and the reference or human-generated summaries. ROUGE is recall-oriented, based on n-gram overlap, and correlates well with human evaluations as discussed by [77] Lin and Hovy. It is a widely used method for assessing quality of automatically generated summaries because it gives high correlation with scores assigned by humans in manually generated summaries..

The evaluation method, in which an automatic summary compared with a reference/manual summary, is based on the n-grams of words of the automatic summary coinciding in the manual summary. [78]. We get a score of recall if the number of co-occurrences are divided by the total *n*-grams of the manual summary, whereas we get a score of precision, if the number of co-occurrences are divided by the total *n*-grams of the automatic summary. These two scores can be further combined to obtain an *F*-score for the automatically generated summary. We can say that the recall tells how much relevant information is obtained from the manual summary, and the precision depicts how much relevant information we get in the automatically generated summary. ROUGE scores range between 0 and 1, where 1 is better.

We calculate the scores for ROUGE-N as follows:

$$Recall = \frac{n - gram(system\ generated\ summary, reference\ summary)}{number\ of\ n\ gram\ in\ reference\ summary}$$

Equation 5.5

$$Precision = \frac{n - gram(system\ generated\ summary, reference\ summary)}{number\ of\ n\ gram\ in\ system\ generated\ summary}$$

Equation 5.6

$$F - score = \frac{2RecallPrecision}{Recall + Precision}$$

Equation 5.7

We have chosen ROUGE-1 as according to Lin [77], for concise summaries ROUGE-1 may suffice.

We have used the DUC 2002 corpus for the evaluation of our approach. The corpus contains different sets of newspaper articles. This data set provides us the test text documents as well as their manual summaries which we use as reference summaries for evaluation. The manually-created extracts are used to train SVM classifier using python on DUC 2002 dataset. We have used polynomial kernel of second degree with a value of

C as 0.0001. We have taken 500 random positive and negative example sentences from these datasets with positive examples indicating the presence of sentence in the extractive summary and negative sentence indicate absence of sentence in the extractive summary. We have used a random collection of documents apart from the training dataset from this corpus to evaluate our results.

We have compared our system to the baseline summaries which are first ten sentences of the text document whose summary is to be generated, reference summaries (human generated), extractive summaries generated by our system without using ontology and the abstractive summaries which are generated after using ontology. We have tested our dataset also on another automatic summarization tool TextRank as given by [66]. We present the evaluation of each system in for each document in our test set in Table 5.1:

Table 5.1: Evaluation of Four Systems on ROUGE

ROUGE-Type	**Task Name**	**System Name**	**Recall**	**Precision**	**F-Score**	**Num Reference Summaries**
ROUGE-1	D103-022	SYSTEM3.TXT	0.4955	0.51402	0.50459	1
ROUGE-1	D103-022	SYSTEM4.TXT	0.41441	0.40351	0.40889	1
ROUGE-1	D103-022	SYSTEM2.TXT	0.56757	0.58333	0.57534	1
ROUGE-1	D103-022	SYSTEM1.TXT	0.63063	0.33333	0.43614	1
ROUGE-1	D105-5959	SYSTEM1.TXT	0.47253	0.43434	0.45263	1
ROUGE-1	D105-5959	SYSTEM4.TXT	0.37363	0.46575	0.41463	1
ROUGE-1	D105-5959	SYSTEM3.TXT	0.34066	0.49206	0.4026	1
ROUGE-1	D105-5959	SYSTEM2.TXT	0.48352	0.43564	0.45833	1
ROUGE-1	D108-111	SYSTEM1.TXT	0.48148	0.33121	0.39245	1
ROUGE-1	D108-111	SYSTEM4.TXT	0.48148	0.34437	0.40154	1
ROUGE-1	D108-111	SYSTEM3.TXT	0.37963	0.41	0.39423	1
ROUGE-1	D108-111	SYSTEM2.TXT	0.35185	0.40426	0.37624	1

ROUGE-1	D114-625-090	SYSTEM1.TXT	0.57843	0.472	0.51982	1
ROUGE-1	D114-625-090	SYSTEM3.TXT	0.35294	0.76596	0.48322	1
ROUGE-1	D114-625-090	SYSTEM2.TXT	0.51961	0.52475	0.52217	1
ROUGE-1	D114-625-090	SYSTEM4.TXT	0.58824	0.5	0.54054	1
ROUGE-1	D109-604	SYSTEM1.TXT	0.69524	0.39891	0.50694	1
ROUGE-1	D109-604	SYSTEM2.TXT	0.52381	0.47009	0.4955	1
ROUGE-1	D109-604	SYSTEM4.TXT	0.68571	0.46452	0.55385	1
ROUGE-1	D109-604	SYSTEM3.TXT	0.34286	0.54545	0.42105	1
ROUGE-1	D102-160	SYSTEM1.TXT	0.56	0.31285	0.40143	1
ROUGE-1	D102-160	SYSTEM3.TXT	0.27	0.25714	0.26341	1
ROUGE-1	D102-160	SYSTEM2.TXT	0.27	0.28421	0.27692	1
ROUGE-1	D102-160	SYSTEM4.TXT	0.43	0.43434	0.43216	1
ROUGE-1	D108-093	SYSTEM3.TXT	0.36364	0.48193	0.41451	1
ROUGE-1	D108-093	SYSTEM4.TXT	0.57273	0.49606	0.53165	1
ROUGE-1	D108-093	SYSTEM2.TXT	0.37273	0.37615	0.37443	1
ROUGE-1	D108-093	SYSTEM1.TXT	0.56364	0.47692	0.51667	1
ROUGE-1	D109-769	SYSTEM4.TXT	0.60784	0.33333	0.43056	1
ROUGE-1	D109-769	SYSTEM2.TXT	0.27451	0.32184	0.2963	1
ROUGE-1	D109-769	SYSTEM3.TXT	0.47059	0.38095	0.42105	1
ROUGE-1	D109-769	SYSTEM1.TXT	0.59804	0.31606	0.41356	1
ROUGE-1	D110-023	SYSTEM1.TXT	0.50495	0.65385	0.56983	1
ROUGE-1	D110-023	SYSTEM2.TXT	0.58416	0.5514	0.56731	1
ROUGE-1	D110-023	SYSTEM4.TXT	0.91089	0.98925	0.94845	1
ROUGE-1	D110-023	SYSTEM3.TXT	0.08911	0.5	0.15126	1
ROUGE-1	D103-119	SYSTEM1.TXT	0.58879	0.33158	0.42424	1
ROUGE-1	D103-	SYSTEM3.T	0.51402	0.33742	0.40741	1

	119	XT				
ROUGE-1	D103-119	SYSTEM2.TXT	0.54206	0.53704	0.53953	1
ROUGE-1	D103-119	SYSTEM4.TXT	0.56075	0.42553	0.48387	1
ROUGE-1	D102-126	SYSTEM1.TXT	0.66981	0.29461	0.40922	1
ROUGE-1	D102-126	SYSTEM4.TXT	0.37736	0.53333	0.44199	1
ROUGE-1	D102-126	SYSTEM2.TXT	0.5	0.5	0.5	1
ROUGE-1	D102-126	SYSTEM3.TXT	0.48113	0.6	0.53403	1
ROUGE-1	D106-070	SYSTEM3.TXT	0.42991	0.66667	0.52273	1
ROUGE-1	D106-070	SYSTEM4.TXT	0.66355	0.58678	0.62281	1
ROUGE-1	D106-070	SYSTEM2.TXT	0.59813	0.60377	0.60094	1
ROUGE-1	D106-070	SYSTEM1.TXT	0.76636	0.36771	0.49697	1
ROUGE-1	D110-095	SYSTEM4.TXT	0.45192	0.4087	0.42922	1
ROUGE-1	D110-095	SYSTEM2.TXT	0.42308	0.36975	0.39462	1
ROUGE-1	D110-095	SYSTEM1.TXT	0.54808	0.43846	0.48718	1
ROUGE-1	D110-095	SYSTEM3.TXT	0.51923	0.34177	0.41221	1
ROUGE-1	D101-185	SYSTEM4.TXT	0.45192	0.39496	0.42152	1
ROUGE-1	D101-185	SYSTEM1.TXT	0.61538	0.27948	0.38438	1
ROUGE-1	D101-185	SYSTEM2.TXT	0.44231	0.40708	0.42396	1
ROUGE-1	D101-185	SYSTEM3.TXT	0.55769	0.41429	0.47541	1
ROUGE-1	D105-244	SYSTEM4.TXT	0.45652	0.30435	0.36522	1
ROUGE-1	D105-244	SYSTEM2.TXT	0.27174	0.30864	0.28902	1
ROUGE-1	D105-244	SYSTEM1.TXT	0.65217	0.30303	0.41379	1
ROUGE-1	D105-244	SYSTEM3.TXT	0.48913	0.48913	0.48913	1
ROUGE-1	D113-981	SYSTEM1.TXT	0.625	0.38462	0.47619	1
ROUGE-1	D113-981	SYSTEM4.TXT	0.625	0.38462	0.47619	1
ROUGE-1	D113-981	SYSTEM3.TXT	0.44231	0.58974	0.50549	1

ROUGE-1	D113-981	SYSTEM2.TXT	0.53846	0.5045	0.52093	1
ROUGE-1	D107-065	SYSTEM1.TXT	0.51515	0.19392	0.28177	1
ROUGE-1	D107-065	SYSTEM2.TXT	0.34343	0.30357	0.32227	1
ROUGE-1	D107-065	SYSTEM3.TXT	0.31313	0.31633	0.31472	1
ROUGE-1	D107-065	SYSTEM4.TXT	0.47475	0.32639	0.38683	1
ROUGE-1	D111-068	SYSTEM4.TXT	0.49515	0.4322	0.46154	1
ROUGE-1	D111-068	SYSTEM2.TXT	0.52427	0.52941	0.52683	1
ROUGE-1	D111-068	SYSTEM3.TXT	0.57282	0.5463	0.55924	1
ROUGE-1	D111-068	SYSTEM1.TXT	0.53398	0.36424	0.43307	1
ROUGE-1	D111-189	SYSTEM3.TXT	0.5567	0.51923	0.53731	1
ROUGE-1	D111-189	SYSTEM4.TXT	0.4433	0.51807	0.47778	1
ROUGE-1	D111-189	SYSTEM2.TXT	0.58763	0.51351	0.54808	1
ROUGE-1	D111-189	SYSTEM1.TXT	0.45361	0.48889	0.47059	1

Here System1.txt denotes extractive summary from our SVM classifier, System2.txt denotes Baseline summaries, System3.txt denotes the summaries generated by TextRank summarizer and System4.txt denotes summaries generated by our proposed system. Number of reference summaries used by each comparison is one here. These scores can be averaged to be shown in the following Table 5.2 for comparison as:

Table 5.2: Comparison of Different Systems

System	**Recall**	**Precision**	**F-score**
Baseline	45.8888	44.8892	45.3091
Textrank	42.0053	48.2547	45.2295
Extractive Summarization by our system	58.1751	37.7685	44.6677
Abstractive summarization our System	52.9745	46.0319	48.5749

As shown in the Table 5.2 our system that generates abstractive summary using ontological structures has higher precision than Baseline summaries. Recall for our system is also higher than both TextRank and Baseline summaries. Using ontological structure with our system generated extractive summaries improves the precision but lowers recall. F-score of our system generating abstractive summaries is highest among baseline, TextRank and extractive summaries. The same is shown using bar graph as in Figure 5.7.

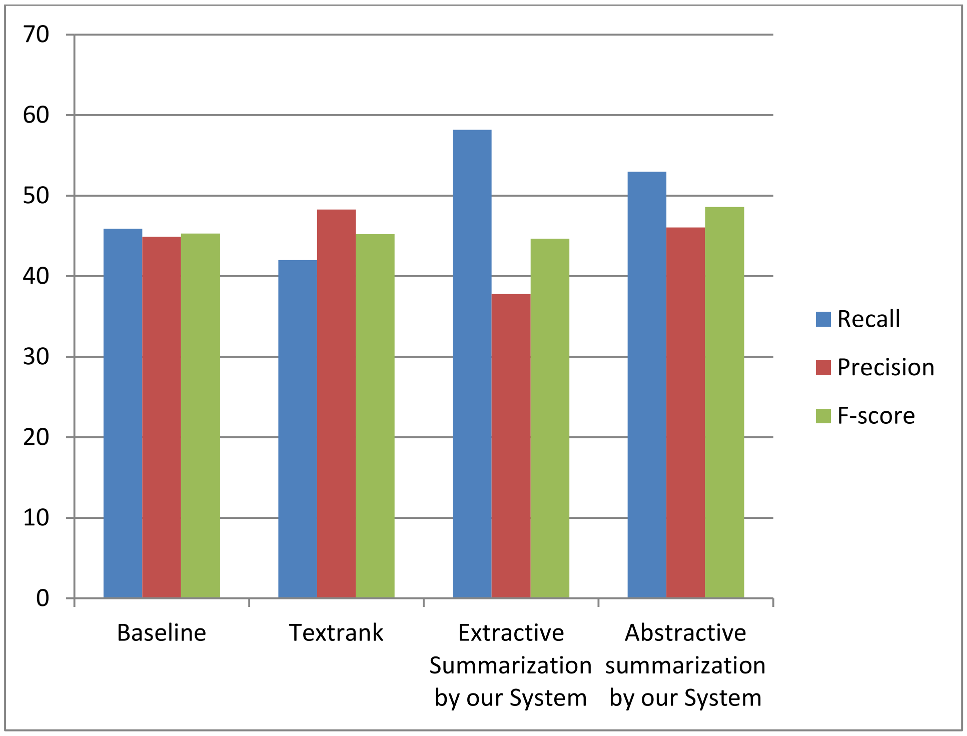


Figure 5.7: Result Analysis of Four Approaches

5.7 CONCLUSION

Summarization of text automatically is a complex task. In our approach, we divided this task into several subtasks. We first extracted the important sentences using SVM classifier trained on DUC 2002 dataset [75] and then condensed the information contained in this extracted sentences further by constructing the ontological graph which relates the sentences from extractive summary semantically and leaves the unnecessary information. This ontology is reworded generating the abstractive summary. We have evaluated our system on news articles from DUC 2002 dataset and compared this system to other systems where competitive performance is shown by our system with better recall than TextRank produced and BaseLine summaries and a better F-score among all four summarization methods.

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSIONS

We commenced our research with exploring the prospects of text understanding so that we can enhance the current web with machine processable capabilities. This kind of enhancement requires intensification of the current web with more machine processable information. This information is a way of linking data between systems or entities that allows for rich, self-describing interrelations of data. Our work is motivated by the need of generating ontologies, as these have become a powerful tool for inter-relating information and hence better text understanding. The in-depth study of literature pointed some glitches (such as unstructured text and fixed set of object properties) in constructing the ontology. We have tried to resolve these issues by proposing and implementing approaches to generate ontology from text and further enriching this ontology. Generation of ontology led to removal of redundancy which further motivated us to propose and implement an approach that uses this ontology for summarizing that text.

Specifically, following are the major contributions by our research work.

Contribution 1: The main challenges involved in constructing ontology from unstructured text has been addressed and resolved as our first contribution. Ontology is generated here using conceptualization and considering semantic roles. Matching the semantic roles of concepts gives an additional feature for efficient merging of sub-ontologies leading to efficient construction of final ontology for better and more correct understanding of text. The rules required for various modules are designed and represented and the approach has been implemented. *We evaluated the performance of the system by looking at the coverage of concepts, their properties and relationships* in the final ontology and comparing it to Open Calais system by Thomson Reuter's which is linked to a market leading ontology extracting entities and for generating *rdf* tagging. It is observed that our system gives better performance than Open Calais as only a fixed set of relationships or object properties are explored by Open Calais. In contrast to that we consider all the non-taxonomical and taxonomical relationships while constructing our ontology.

Contribution 2: The next contribution is the proposal of a technique of enriching ontology by providing extra information to it. The ontology is enriched by providing the class labels for data properties extracted during the generation of ontology. The proposed technique forms a dictionary of adjectives and identifies labels for these adjectives. A total of 16 labels are identified that can be given to the data properties, which are extracted during the process of construction of ontology. These labels are associated with the data properties using a link which is labelled as '*has_class'*. We have implemented this approach and the shown the resulting ontology using GraphViz tool.

Contribution 3: Our ontology is capable of removing redundancy and concision of information as we have removed the un-necessary and redundant words/phrases in the course of construction of ontology. So we have used our ontology to summarize the text as our third contribution. To ease the process of summarization we proposed and implemented a novel approach that used a machine learning tool SVM first and got an extractive summary from a text document. This extractive summary is further used to construct ontology by our proposed and implemented technique. We used this ontology to generate the abstractive summary of the text document. Our approach shows better performance when compared to baseline and another summarizer named as TextRank. We get better recall for our extractive summarization and abstractive summarization. F-score of our system that generates abstractive summary by using ontology is better than Baseline system, TextRank system and the extractive summarization.

6.2 SCOPE FOR FUTURE WORK

Though the present work proposes the complete design of framework to develop the automatic ontology, still no research work is closed solution; means every research work can be further explored and extended. Therefore, following are some of the possible lines in which the present work can further explored or extended in future:

1. **Considering Other Semantic Information**

   The understanding of semantic of a text is vast and complex task involving many aspects that contribute in it. The present work is taking care of semantic roles along with concepts, relations and properties. The proposed system can be further enriched by identifying other semantic aspects in the text such as context of the event occurred and the objects participating.

2. **Ontology Integration to Linked data**

   The proposed Ontological framework can be integrated to Linked data such as DBpedia for providing context to concepts so that it can be improved further.

3. **Ontology Driven Information System**

   The proposed system is providing enriched information; this can be further used into an information system for increasing the quality of search results. This information can also be used for opinion mining.

4. **Cross Validating Labels of Data Properties**

   The labels given to data properties during enrichment of ontology (in our current proposal) can be cross verified using some additional mechanism based on standard lexical database such as WordNet.

REFERENCES

[1] Tim Berners-Lee, James Hendler And Ora Lassila, "The Semantic Web*"., Scientific American: Feature Article,* May 2001.

[2] Na Xue, Suling Jia, Jinxing Hao and Qiang Wang "Scientific ontology construction based on interval valued fuzzy theory under Web 2.0," *Journal of Software, 8(8)*, pp.1835-1842. , 2013.

[3] Ivana Luksova, "Ontology Enrichment Based on Unstructured Text Data," *Masters Thesis*, 2013.

[4] Silvia Calegari and Gabriella Pasi, "Personal ontologies: Generation of user profiles based on the yago ontology," *Generation of Information processing & management, vol. 49, no. 3,* pp. 640–658, 2013.

[5] Sun Yu, Li Zhipping, "Ontology-based domain knowledge representation,", *"Ontology-base 4th International Conference on Computer Science and Eucation*, pp. 174-177, 2009

[6] John Barnden, Jizheng Wan, "A New Semantic Model for Domain-Ontology Learning, ," *Springer-Verlag Berlin Heidelberg*, 2011.

[7] Antonio De Nicola and Michele Missikoff, "A Lightweight Methodology for Rapid Ontology Engineering," *ACM VOL. 59 , NO. 3* , pp. 79-86, 2016.

[8] Trung-Kien Tran, Robert Meersman and Christophe Debruyne, "Grounding Ontologies with Social Processes and Natural Language," *Springer-Verlag Berlin Heidelberg, J Data Semant 2*, pp. 89–118, 2013.

[9] Asuncio´n Go´mez-Pe´rez, Mariano Ferna´ndez-Lo´pez, Mari Carmen Sua´rez-Figueroa, "The NeOn Methodology for Ontology," *Ontology Engineering in a Networked World,Springer-Verlag Berlin Heidelberg*, 2012.

[10] Saira Andleeb Gillani, "From text mining to knowledge mining: An integrated framework of concept extraction and categorization for domain ontology," *Budapesti*

Corvinus Egyetem, 2015.

[11] Abel Browarnik, Oded Maimon, "Ontology Learning from Text: Why the Ontology Learning Layer Cake is not Viable," *Int. J. Signs Semiot. Syst 4(2)*, pp. 1-14, 2015.

[12] Abdul Malik, Al-Salman Abeer Al-Arfaj, "Ontology Construction from Text: Challenges and Trends," *International Journal of Artificial Intelligence and Expert Systems (IJAE), Volume (6) : Issue (2)*, 2015.

[13] Jan Daniel Bothma, "Ontology learning from Swedish text," in *International Conference on Computer Science and Education*, 2010.

[14] Khurshid Ahmad and Lee Gillam, "Automatic Ontology Extraction from Unstructured Texts," in *On the move to meaningful Internet*, 2005.

[15] Douwe Kiela, and Maximilian Nickel Stephen Roller, "Hearst Patterns Revisited:Automatic Hypernym Detection from Large Text Corpora," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, Melbourne, Australia, pp. 358–363, 2018.

[16] Kidane Woldemariyam, Fekade Getahun, "Integrated Ontology Learner: Towards Generic Semantic Annotation Framework," in *MEDES'17*, Bangkok, Thailand., November 7–10, 2017.

[17] Harith Alani, "Position Paper: Ontology Construction from Online Ontologies," *WWW2006*, May 22–26, 2006.

[18] Junli Li, Zongyi He and Qiaoli Zhu, "An Entropy-Based Weighted Concept Lattice for Merging Multi-Source Geo-Ontologies," *doi:10.3390/e15062303*, pp. 2303-2318, 2013.

[19] C.P., & Sumathi, V.P. Abinaya, "Semi-Automatic Ontology Merging of Domain Specific Ontologies," *International Journal of Science and Research (IJSR) ISSN (Online): 2319-7064* , 2013.

[20] Andreia Dal Ponte Novelli and José Maria Parente de Oliveira, "Simple Method for Ontology Automatic Extraction from Documents," *International Journal of*

Advanced Computer Science and Applications(IJACSA), Volume 3 Issue 12, pp. DOI: 10.14569/IJACSA.2012.031206, 2012.

[21] Guntars Bumans, "Mapping between Relational Databases and OWL Ontologies: an Example," *Computer Science and Information Technologies, Scientific Papers, University of Latvia*, p. Vol. 756, 2010.

[22] Jean Vincent Fonou-Dombeu Kgotatso Desmond Mogotlane, "Automatic Conversion of Relational Databases into Ontologies: A Comparative Analysis of Protégé Plug-ins Performances," *International Journal of Web and Semantic Technology (IJWesT)*, vol. arXiv:1611.02816, 2016.

[23] Marti A. Hearst, "Automatic Acquisition of Hypernyms from Large Text Corpora," in *PRoc. of Collins*, Nantes, 1992.

[24] Bernardo Magnini Hristo Tanev Tanev, "Weakly Supervised Approaches for Ontology Population," 2008.

[25] Xing Jiang and Ah-Hwee Tan, "CRCTOL: A semantic-based domain ontology learning system," *Journal of the American Society for Information Science and Technology, Volume 61, Issue 1*, pp. 150–168, January 2010.

[26] Kalliopi Zervanou2, and Euripides G.M. Petrakis, C.J. Hopfe et al. (Eds.) Drymonas1, "Unsupervised Ontology Acquisition from Plain Texts: The OntoGain System ," *NLDB 2010, LNCS 6177, Springer-Verlag Berlin Heidelberg*, pp. 277–287, 2010.

[27] Hoifung Poon and Pedro Domingos, "Unsupervised Ontology Induction from Text,".

[28] G. Zayaraz G. Suresh kumar, "Concept relation extraction using Naı̈ve Bayes classifier for ontology-based question answering systems," *Journal of King Saud University – Computer and Information Sciences*, pp. 13–24, 2015.

[29] Andreas Scheuermann, Stephan Bloehdorn Julia Hoxha, "An Approach to Formal and Semantic Representation of Logistics Services," in *In Proceedings of the ECAI'10 Workshop on Artificial Intelligence and Logistics* pp. 73-78, 2010.

[30] Aditya Mishra, William W. Cohen Bhavana Dalvi, "Hierarchical Semi-supervised Classification with Incomplete Class Hierarchies," in *WSDM'16*, San Francisco, CA, USA, 2016.

[31] Marius-Gabriel, "Semantically Enriching Content Using OpenCalais ," *Thomsan Reuters*, 2016.

[32] Amit Kumar Dhar and O. P. Vyas Monika Rani, "Ontology Learning Based on Topic Modeling," *Semi-Automatic Terminology*, 2017.

[33] Anna Lisa Gentile, Daniel Gruhl, Kenneth Clarkson, "User-Centric Ontology Population," in *eswc-conferences,* 2018.

[34] P. Danielsen and S. Afroz K. Arabshian, "LexOnt: A Semi-Automatic Ontology Creation Tool for Programmable Web," in *AAAI Spring Symposium: Intelligent Web Services Meet Social Computing*, pp.2-8, 2012.

[35] Ankur Padia, Gaurav Maheshwari, Priyansh Trivedi, Jens Lehmann and Sourish Dasgupta, "Formal Ontology Learning from English IS-A Sentences," *arXiv:1802.03701v1 [cs.AI]* , 11 Feb 2018.

[36] D. Gasevic, M. Hatala. A. Zouaq, "Towards open ontology learning and filtering ," *Information Systems, vol. 36, no.7,* pp. 1064–1081, 2011.

[37] Yannick Toussaint, Amedeo Napoli. Rokia Bendaoud, "PACTOLE: A methodology and a system for semi-automatically enriching an ontology from a collection of texts," in *16th International Conference on Conceptual Structures ICCS'08* , Toulouse, France, pp. 203-216, 2008.

[38] R. Navigli and P. Velardi., "Ontology enrichment through automatic semantic annotation of on-line glossaries.," in *In 15th International Conference in Knowlegde Engineering and knowledge Management (EKAW 2006),* Czech Re-public, pages 126–140, 2006.

[39] A. Ferrara, and G.N. Hess S. Castano, "Discovery-driven ontology evolution," in *3rd Italian Semantic Web Workshop,* Pisa, Italy, 2006.

[40] Armando Stellato Maria Teresa Pazienza, "Linguistic Enrichment Of Ontologies: a methodological framework," in *AI Research Group, DISP*, University of Rome, Tor Vergata, 2006

[41] J., Milic-Frayling, N., Grobelnik, M Leskovec, "Extracting Summary Sentences Based on the Document Semantic Graph," in *MSR-TR-2005-07*, January 31, 2005.

[42] E. Canhasi, *Graph-based models for multi-document summarization*. Ljubljana: PhD thesis, 2014.

[43] D., & Strube M Parveen, "Integrating Importance, Non-Redundancy and Coherence in Graph-Based Extractive Summarization," in *on", Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*

[44] Prasenjit Mitra, Kazunari Sugiyama Siddhartha Banerjee, "Multi-Document Abstractive Summarization Using ILP Based Multi-Sentence Compression," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 2015.

[45] R. & Sivaranjani, N Ragunath, "Ontology Based Text Document Summarization System,Using Concept Terms," *ARPN Journal of Engineering and Applied Sciences* , 2006.

[46] L.,Umbrath, W., & Wetzker, R Hennig, "An Ontology-based Approach to Text Summarization.," in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and International Conference on Intelligent Agent Technology - Workshops*, Sydney, NSW, Australia, 9-12 December, 2008.

[47] R., Chen, P.,& Lu, W. Verma, "A Semantic Free-text Summarization System Using Ontology Knowledge," *IEEE Transactions on Information Technology in Biomedicine. Vol. 5, No. 4*, pp. 261- 270, 2009.

[48] E., Cagliero, L., & Jabeen, S Baralis, "Multi-document summarization based on the Yago ontology," *Expert Systems with Applications 40(17)*, 2013.

[49] Reddy Y.S.& Kumar A.P.S, "An Efficient Approach for Web document

summarization by Sentence Ranking," *International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7*, July 2012.

[50] M. S., Bewoor, M. S., & Patil, S. H Patil, "A Hybrid Approach for Extractive Document Summarization Using Machine Learning and Clustering Technique. (IJCSIT) ," *International Journal of Computer Science and Information Technologies, Vol. 5 (2)* , 2014.

[51] S.,& Patilb, P Babara, "Improving Performance of Text Summarization," *Procedia Computer Science 46, ScienceDirect*, pp. 354 – 363, 2015.

[52] Nedunchelian Ramanujam and Manivannan Kaliappan, "An Automatic Multidocument Text Summarization Approach Based on Naïve Bayesian Classifier Using Timestamp Strategy," *The Scientific World Journal, Volume 2016*.

[53] S., Kumar, A., Mangal, A., Singhal, S Singh, "Bilingual Automatic Text Summarization Using Unsupervised Deep Learning. ," in *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)* , 2016.

[54] Min-Yen Kan, Tat-Seng Chua Long Qiu, "A Public Reference Implementation of the RAP Anaphora Resolution Algorithm", 2004

[55] M.C.D., & Manning, C.D Marneffe, "Stanford dependency parser," *[Computer Software] Retrieved from nlp.stanford.edu/software/stanford-dependencies*, 2015.

[56] Pavel Kuksa Ronan Collobert, "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research 12* , pp. 2461-2505, 2011.

[57] Eleftherios Koutsofios & Stephen North Emden Gansner, "Drawing graphs with dot," *,Technical Report, AT&T Research. URL http://www.graphviz.org/Documentation/dotguide.pdf*, 2010.

[58] C. Cortes and V. Vapnik, "Support vector networks.," *Machine Learning*, pp. 20:273–297, 1995.

[59] Joachims T., "Text categorization with Support Vector Machines: Learning with many relevant features," *Nédellec C., Rouveirol C. (eds) Machine Learning: ECML-*

98. ECML Lecture Notes in Artificial Intelligen, pp. vol 1398. Springer, Berlin, Heidelberg, 1998.

[60] S. Patwardhan & J. Michelizzi T. Pedersen, "WordNet:Similarity - Measuring the Relatedness of Concepts," in *In Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics Demonstrations*, Boston, 2004.

[61] Daniel Jurafsky & James H. Martin, "Speech and Language Processing," Book chapters, 2015.

[62] Krzysztof Janowicz, "Extending Semantic Similarity Measurement with Thematic Roles GeoS'05," in *Proceedings of the First international conference on GeoSpatial Semantics*, 2005.

[63] Miriam Connor, Natalia Silveira, Samuel R. Bowman, Timothy Dozat and Christopher D. Manning Marie-Catherine de Marneffe, "More constructions, more genres: Extending Stanford Dependencies," 2013.

[64] Geotge A. Miller, "WordNet: A Lexical Database for English," *Communication of the ACM Vol. 38, No. 11:39-41*. 1995

[65] David Hall, James R. Curran and Dan Klein Jonathan K. Kummerfeld, "Parser Showdown at the Wall Street Corral: An Empirical Investigation of Error Types in Parser Output," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Jeju Island, Korea, 12–14 July 2012.

[66] R., & Tarau, P Mihalcea, "Textrank: Bringing order into text," in *Proceedings of EMNLP* , Barcelona, Spain, pp. 404–411, 2004

[67] Naomie Salim, Haleem Farman Atif Khan, "Clustered genetic semantic graph approach for multi-document abstractive summarization," in *International Conference on Intelligent Systems Engineering (9ICISE),* Los Angeles, USA, 2016.

[68] I. F. Moawad and M. Aref, "Semantic graph reduction approach for abstractive Text

Summarization," in *Seventh International Conference on Computer Engineering & Systems (ICCES)*, Cairo, Egypt, 2012.

[69] Henrik Danielsson, Arne Jönsson Christian Smith, "A More Cohesive Summarizer," in *Proceedings of COLING 2012: Posters*, Mumbai, December 2012.

[70] Mridusmita Sharma and Kandarpa Kumar Sarma, *Soft-Computational Techniques and Spectro-Temporal Features for Telephonic Speech Recognition: An Overview and Review of Current State of the Art*. Gauhati University, India: Handbook of Research on Advanced Hybrid Intelligent Techniques and Applications, 2016.

[71] Thorsten Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," in *LS VIII Number 23, University of Dortmund, ,* 1997.

[72] Mohamed Fattah,Fuji Ren Nadira Begum, "Automatic text summarization using support vector machine," in *International journal of innovative computing, information & control: IJICIC 5(7),* july 2009.

[73] Shivakumar. Soumya, "Text summarization using clustering technique and SVM technique," *International Journal of Applied Engineering Research*, ISSN 0973-4562, Volume 10, no. 12 , pp. 28873-28881, 2015.

[74] Hideki Isozaki, Eisaku Maeda Tsutomu Hirao, "Extracting Important Sentences with Support Vector Machines," *Proceedings of the 19th intenational conference on Computational Liguistics- Volume 1, 1-7,* 2002

[75] P. Over, W. Liggett " Introduction to DUC: An Intrinsic Evaluation of Generic News Text Summarization Systems ", *In Conjunction with the ACL 2002 and including the DARPA/NIST sponsored DUC 2002 Meeting on Text Summarization,* National Institute of Standards and Technology, 2002

[76] Dang H., and Harman D Over P., "DUC in Context," *Information Processing and Enhanced Graph Based Approach for Multi Document Summarization 341 Management, vol. 43, no. 6,* pp. 1506-1520, 2007.

[77] C.-Y., Hovy, E. Lin, "Automatic evaluation of summaries using n-gram co-occurrence statistics," in *NAACLHLT-2003*, 2003.

[78] Socorro Gama-Castro, Citlalli Mejía-Almonte, Marco-Polo Castillo-Villalba, Luis-José Muñiz-Rascado, Julio Collado-Vides Carlos-Francisco Méndez-Cruz, "First steps in automatic summarization of transcription factor properties for RegulonDB: classification of sentences about structural domains and regulated processes," 2017.

[79] U M Fayyad, A Wierse, and G G Grinstein, *Information visualization in data mining and knowledge discove,*: Morgan Kaufmann, 2002.

[80] Garcia-Molina, Junghoo Cho, and Hector, "Estimating frequency of change," *VLDB 2000, Research track*, 2000.

[81] Staab, Marc Ehrig, and Steffen, "QOM - Quick Ontology Mapping," *In: McIlraith S.A, The Semantic Web – ISWC 2004. Lecture Notes in Computer Science, vol 3298. Springer, Berlin, Heidelberg,* 2004

[82] W., W. Liu, and M. Bennamoun Wong, "Ontology learning from text: A look back and into the future ," *ACM Computing Surveys (CSUR),*pp. 44(4) , 2012.

[83] Abel Browarnik and Oded Maimon, "Ontology Learning from Text Departing the Ontology Layer Cake," in *ALLDATA 2015 : The First International Conference on Big Data, Small Data, Linked Data and Open Data*, 2015.

[84] M. Shamsfard and A. Barforoush, "Learning ontologies from natural language texts," *Int. J. Human-Computer Studies*, pp.17–63, 2014.

[85] S. Saad, R. Abood, M. Shakir M. Sheker, "Domain-Specific Ontology-Based Approach For Arabic Question Answering," *Journal of Theoretical And Applied Information, E-ISSN 1817-3195 / ISSN 1992-8645). Vol 83*, 2016.

[86] N.A. and D.Y. Turdakov Astrakhantsev, "Automatic construction and enrichment of informal ontologies: A survey," *Programming and Computer Software 39(1)*, pp. 34-42, 2013.

[87] J., J. Liu, and X. Wang Zhang, "Simultaneous Entities and Relationship Extraction

from Unstructured Text," *International Journal of Database Theory and Application, 2016. 9(6)*, 151-160, 2016.

[88] Jose Camacho-Collados, "Why we have switched from building full-fledged taxonomies to simply detecting hypernymy relations," *arXiv*, 2017.

[89] Michael Cochez, Usman Qamar Bushra Zafar, "Using Distributional Semantics for Automatic Taxonomy Induction," in *Proceedings of semeval task 13*, pp. 1320–1327, 2016.

[90] Horacio Saggion, Francesco Ronzano and Roberto Navigli Luis Espinosa-Anke, "ExTaSem! Extending, Taxonomizing and Semantifying Domain Terminologies," *Association for the Advancement of Artificial*, 2016.

[91] Jin Liu, Xiaofeng Wang, Jin Wang Qiuxia Song, "A Novel Automatic Ontology Construction Method Based on Web Data," in *Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2014.

[92] Rosario Girardi, Carla Faria Jone Correia, "Extracting Ontology Hierarchies From Text" Conference: Proceedings of the 23rd International Conference on Software Engineering & Knowledge Engineering (SEKE'2011), Eden Roc Renaissance, Miami Beach, USA, July 7-9, 2011

.

BRIEF PROFILE OF RESEARCH SCHOLAR

Amita Arora is pursuing her Ph.D in Computer Engineering from J.C. Bose University of Science and Technology YMCA, Faridabad. She did her M.Tech. (Computer Engineering) from YMCA University of Science and Technology in year 2008, and B.Tech.(Information Technology) from Kurukshetra University, Kurukshetra in 2003. Ms. Amita Arora has over 13 years of experience in teaching B.Tech, MCA and M.Tech courses. Her area of interests includes Analysis and Design of Algorithms, Computer Graphics, Programming Languages (C, C++, .Net etc.), Data Structures and Internet & Web Technologies. She has published research papers in various journals and conferences of international fame. Currently, she is working as Assistant Professor in the Department of Computer Engineering at J.C. Bose University of Science and Technology YMCA, Faridabad.

LIST OF PUBLICATIONS

Sr. No	Title of the Paper	Journal/Conference	Year Month Vol(Issue)	Page No.	ISSN No.	Indexing Listing
1	Automatic Ontology Construction using Conceptualization and Semantic Roles	International Journal of Information Retrieval Research	2017/ Sept. 7(3)	62-80	2155-6377	ESCI, UGC
2	Machine Learning Approach for Text Summarization	International Journal of Database Theory and Application	2017/ April 10(8)	83-90	2005-4270	Scopus/ UGC
3	A Novel Hybrid Approach for Extracting Relations from	National Conference on Role of Science and Technology Towards, Make In	2016/ March			

	Plain Text	India, YMCAUST				
4	Document Summarization Techniques: A Review	Recent Trends in Computing and Communication technologies (RCCT-2016), DCRUST	2016/ Sept.	98-101	ISBN: 978-93-86256-02-7	
5	Extractive Summarization using Support Vector Machines	National Conference on Advances In Mathematics & Computing (AMC-2017)	2017/ March			