Addis Ababa University

College of Natural Sciences

Generic Semantic Annotation Framework with Integrated Ontology Learner

Kidane Woldemariyam

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

November 2017

Addis Ababa University

College of Natural Sciences

Kidane Woldemariyam

Advisor: Fekade Getahun (PhD)

This is to certify that the thesis prepared by Kidane Woldemariyam, titled: *Generic Semantic Annotation Framework with Integrated Ontology Learner* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

	Name	**Signature**	**Date**
Advisor:	Dr. Fekade Getahun		
Examiner:	Dr. Dida Midekso		
Examiner:	Dr. Yaregal Assabie		

Abstract

The Web has become a source of information, where information is provided by humans for humans and its growth has increased the necessity to get solutions that intelligently extract valuable knowledge from existing and newly added web documents with no (minimal) supervisions. However, due to the unstructured nature of existing data on the Web, effective extraction of this knowledge is limited for both human beings and software agents. Thus, our research goal is to design a generic framework that automatically learns ontology from unstructured text and annotate web documents semantically using ontology as a semantic repository. This allows software agents in various fields such as knowledge management, expert systems, and semantic web to understand and process web resources semantically. The proposed framework has the following distinctive features: (1) three granularity level of document tagging (word, sentence and paragraph); (2) structure, language and domain awareness; (3) generic ontology learner integration; (4) annotation maintenance; (5) annotation verification and (6) artificial neural network approaches adoption. We experiment the feasibility of the proposed approach using Amharic news collected from Walta news agency and Amharic Wikipedia. Our result of experimentation shows that the proposed solution exhibits 70.68% of precision, 66.89% of recall and 68.53% of f-measure in semantic annotation for a morphologically complex Amharic language with a limited size dataset. Our experiments demonstrate that the proposed solution has the capability to provide domain and language independent semantic annotation except of general patterns used for ontology learning refinements. Our solution significantly reduces manual annotation and learning cost used for both semantic annotation and ontology learning of web documents with its nature of adaptability with minimal modification. The results have also implied that neural network techniques are promising for both semantic annotation and ontology learning, especially for less resourced languages in comparison to language dependent techniques that have cost of speed and challenge of adaptation into new domains and languages.

Keywords: Information Extraction, Knowledge Base, Ontology Learning, Semantic Annotation, Semantic Understanding, Semantic Web, Artificial Neural Networks.

Dedication

To my Great Grandmother (እማማ)

To my Grandfather (አባዬ)

Acknowledgements

Thanks to Almighty God and St. Virgin Mary for the completion of this thesis. I would like to express my sincere gratitude to my advisor Dr. Fekade Getahun for the perfect balance of guidance and freedom he gave me throughout this research work. The door to Dr. Fekade office was always open whenever I had a question, and he helped me a lot to clarify the research issues and gave me a big instruction for my academic life with his motivational, scholarly, and scientific advice. He also gave me a lot of research related resources, including articles, books, and scientific tools which helped me a lot to understand and experiment different problems related to this research work. In many ways, he has shown me the path for how to know, understand, and apply computer science and related fields in recent research areas with his inspiring encouragements.

Besides my advisor, I would like to thank all of my teachers during my stay at Addis Ababa University for their teaching that helped me to solidify my thinking on many topics related to this research work. Their professional working attitudes also greatly influenced me, from which I gain benefits and experiences for my future study and teaching. I would also like to thank the examining committee: Dr. Dida Midekso and Dr. Yaregal Assabie for their encouragement and insightful comments.

Next, I would also like to acknowledge Dr. Sefu Macea Amharic language and literature instructor at Bahir Dar University, and PhD students of School of Linguistic at Addis Ababa University: Ato Yirgalem Girma, Ato Zinawork Assefa, Ato Mengistu Tadesse and W/ro Helen Efrem for their linguistic related supports including gold standard preparation, semantic annotation, and ontology learning model evaluation. I am gratefully indebted for their valuable linguistic comments and suggestions.

Finally, I express my very profound gratitude to my family, colleagues, and friends. I am thankful to my family for their loving, considerations and great confidence in me all through my life. Thank you Mam, Dad, and Mahe for all of your support and this accomplishment would not have been possible without you and the rest of my family.

Table of Contents

List of Tables v
List of Figures vi
List of Algorithms viii
Acronyms and Abbreviations ix
1. Introduction 1
1.1 Background 1
1.2 Motivation 3
1.3 Statement of the Problem 4
1.4 Objectives 6
1.5 Methods 6
1.6 Scope and Limitations 7
1.7 Application of Results 8
1.8 Thesis Organization 8
2. Literature Review 9
2.1 Introduction 9
2.2 Natural Language Processing 9
2.2.1 Feature Representation 12
2.2.2 Word Maps and Language Models 12
2.3 Information Extraction 13
2.3.1 Structure Analysis 14
2.3.2 Information Extraction Approaches used in Semantic Annotation 15
2.3.3 Data (Text) Mining 17
2.3.4 Named Entity Recognition 18
2.3.5 Relation Extraction 19
2.4 Semantic Understanding 19
2.5 Machine Learning 23
2.5.1 Supervised Learning 24
2.5.2 Unsupervised Learning 24
2.5.3 Reinforcement Learning 25
2.6 Neural Networks 25

2.7 Semantic Web 27
2.8 Semantic Annotation 29
2.8.1 Semantic Annotation Standards 31
2.8.2 Semantic Annotation Requirements 31
2.8.3 Semantic Annotation Phases 32
2.8.4 Semantic Annotation Approaches 34
2.8.5 Semantic Annotation Challenges 36
2.9 Ontologies 36
2.9.1 Ontology Graph 37
2.9.2 Ontology Modeling 39
2.9.3 Ontology Learning 40
3. Related Work 44
3.1 Introduction 44
3.2 Semantic Annotation Frameworks 44
3.2.1 BNOSA 44
3.2.2 Annotea 45
3.2.3 CREAM 46
3.2.4 Arabic Semantic Annotation Framework 47
3.3 Pattern based Semantic Annotations 47
3.3.1 KIM 48
3.3.2 SemTag 49
3.3.3 ONTEA 50
3.3.4 OntoAnnotate 51
3.3.5 PANKOW 51
3.3.6 Automatic Ontology based Annotation for Arabic Web Content 52
3.4 Machine Learning based Semantic Annotations 53
3.4.1 MnM 53
3.4.2 Semantically Annotating Corpus using Machine Learning 54
3.4.3 SARM 54
3.5 Multi Strategy based Semantic Annotation 55
3.6 Requirements 56

3.7 Summary 57
4. Generic Semantic Annotation Framework 58
4.1 Introduction 58
4.2 The Proposed Framework 59
4.2.1 Document Analysis 59
4.2.2 Semantic Annotation 64
4.2.3 Ontology Learning 77
4.3 Semantic Annotation Workflow Summary 92
5. Experiment 94
5.1 Introduction 94
5.2 Experimental Setup 94
5.2.1 Datasets 94
5.2.2 Packages and Tools 95
5.2.3 User Interface 96
5.2.4 Experimental Setting 98
5.3 Experiments 98
5.3.1 Document Analysis 99
5.3.2 Word Embeddings 100
5.3.3 Ontology Learning 102
5.3.4 Semantic Annotation 105
5.4 Evaluation 106
5.4.1 Ontology Learning Evaluation 106
5.4.2 Semantic Annotation Evaluation 108
5.5 Discussion 109
6. Conclusions and Future Works 111
6.1 Conclusions 111
6.2 Future Works 113
References 114
Annexes 126
Annex A - Related Works and Approaches Comparison 126
Annex B - The LangID's 97 Languages 128

Annex C - Sample Domain Categorization Training Dataset 128
Annex D - Sample Indexed Word List 129
Annex E - Indexed NE Tagsets 129
Annex F - Sample IOB Encoded NER Dataset 130
Annex G - Sample Extracted Single Terms with TFIDF 131
Annex H - Sample Ontology with Sample Taxonomy Induction Steps 132
Annex I - Sample Source Code 137
Annex J - Ontology Learner Evaluation Questionnaire's Template 140
Annex K - Manual Semantic Annotation Questionnaire's Template 141

List of Tables

Table 2.1: *Document Metadata and Structure Template Elements* 14
Table 2.2: *Relationship Pairs in Word Embeddings from English word2vec Model* 21
Table 2.3: *Summary of ML Categories* 25
Table 2.4: *Examples of Different Annotation Strategies* 35
Table 4.1: *NE Tagsets with Examples* 67
Table 4.2: *Extracted Content Refinement Metrics* 70
Table 4.3: *Verification Scoring Metrics* 75
Table 4.4: *Candidate Concept to Domain Contingency Table* 83
Table 4.5: *Semantic Concept Relations* 85
Table 5.1: *Statistics of the Corpora* 94
Table 5.2: *Description of Packages and Tools* 95
Table 5.3: *Overview of the Experiments* 98
Table 5.4: *Statistics of News Articles used for Domain Categorization* 100
Table 5.5: *Word embeddings Training Settings* 101
Table 5.6: *Nearest Neighbors in the Word Embedding Space* 101
Table 5.7: *Sample Extracted Single Term Concepts with help of Wikipedia Infobox* 102
Table 5.8: *Sample Extracted Multi Word Concepts* 102
Table 5.9: *Sample List of Patterns used for Taxonomic Relation Refinement* 103
Table 5.10: *Sample Result for Correlation based Non-Taxonomic Extraction* 104
Table 5.11: *Sample List of Patterns for Meronyms (part-of) Relation* 104
Table 5.12: *Sample List of Patterns for Holonym (has-a) Relation* 105

List of Figures

Figure 1.1: *Web Traffic Analysis Report by Incapsula* 4
Figure 2.1: *Typical stages in the Information Extraction Process* 13
Figure 2.2: *Machine Learning Steps using Train and Test Dataset* 23
Figure 2.3: *A Venn diagram showing learning and AI dependency* 23
Figure 2.4: *Diagram of Recurrent Neural Networks* 26
Figure 2.5: *Semantic Web Building Blocks* 28
Figure 2.6: *Example for Content to Concept Mapping in Semantic Annotation* 30
Figure 2.7: *Summary of Semantic Annotation Approaches* 34
Figure 2.8: *Knowledge Components of Ontology Learning from Text* 38
Figure 3.1: *Components of BNOSA Framework* 45
Figure 3.2: *Architecture of Annotea* 45
Figure 3.3: *Architecture of CREAM* 46
Figure 3.4: *Framework for Arabic Document Semantic Annotation* 47
Figure 3.5: *KIM Semantic IE Flow Diagram* 48
Figure 3.6: *SemTag Architecture* 50
Figure 3.7*: OntoAnnotate Semantic Annotation Environment* 51
Figure 3.8: *The Process of PANKOW* 52
Figure 3.9: *Annotation Process for Arabic Web Content* 53
Figure 3.10*: Architecture of the Korean Document Semantic Annotation* 55
Figure 4.1: *Proposed Framework for Web Documents Semantic Annotation* 58
Figure 4.2: *Document Preprocessing Workflow* 62
Figure 4.3: *Domain and Privacy Tagging Workflow* 64
Figure 4.4: *Document Information Extraction Workflow* 66
Figure 4.5: *LSTM Components Interaction for NER as Term Level Extraction* 67
Figure 4.6: *Sequence Tagging for NER* 67
Figure 4.7: *Content to Concept Mapping and Verification Workflow* 74
Figure 4.8: *Ontology Learning Architecture* 78
Figure 4.9: *Corpus Preprocessing Workflow* 79
Figure 4.10: *Concept Extraction Workflow* 79
Figure 4.11: *Taxonomic Induction Workflow* 86

Figure 4.12: *Semantic Annotation Workflow Summary* 93
Figure 5.1: *Amharic Wikipedia Dump Preprocessing Result* 95
Figure 5.2: *Homepage User Interface for Amharic Semantic Annotator* 97
Figure 5.3: *Sample User Interfaces* 97
Figure 5.4: *Tourism Domain Word2Vec Model Visualization using t-SNE* 101
Figure 5.5: *Sample Tourism Domain Ontology* 103
Figure 5.6: *Ontology Learner Execution Time Series* 107
Figure 5.7: *Comparison of Existing and Our Ontology Learner* 107
Figure 5.8: *Term Extraction Evaluation Result* 108
Figure 5.9: *Semantic Annotation Performance Evaluation* 109

List of Algorithms

Algorithm 4.1: Structure Analysis Pseudocode 60
Algorithm 4.2: Redundancy Checking Pseudocode 61
Algorithm 4.3: Document Normalization Pseudocode 61
Algorithm 4.4: Domain Categorizer Modeling Pseudocode 62
Algorithm 4.5: Knowledge Oriented Document Privacy Tagging Pseudocode 63
Algorithm 4.6: Higher Level Semantic Annotation Pseudocode 65
Algorithm 4.7: Term Level Extraction Pseudocode 68
Algorithm 4.8: Sentence Level Extraction Pseudocode 69
Algorithm 4.9: Paragraph Level Tagging Pseudocode 71
Algorithm 4.10: Content to Concept Mapping Pseudocode 71
Algorithm 4.11: Semantic Annotation Verification Pseudocode 73
Algorithm 4.12: Semantic Annotation Persistency Pseudocode 76
Algorithm 4.13: Semantic Annotation Maintenance Pseudocode 77
Algorithm 4.14: Concept Extraction Pseudocode 80
Algorithm 4.15: Concept to Domain Mapping Pseudocode 83
Algorithm 4.16: Taxonomic Relation Extraction Pseudocode 87
Algorithm 4.17: Cluster Labeling Pseudocode 88
Algorithm 4.18: Compute Label Vote Pseudocode 89
Algorithm 4.19: Correlation based Non-taxonomic Relation Extraction Pseudocode 91
Algorithm 4.20: Word2vec Analogy based Relation Extraction Pseudocode 92

Acronyms and Abbreviations

AE:	Above Expectation
AI:	Artificial Intelligence
ANN:	Artificial Neural Networks
BNOSA:	Bayesian Network and Ontology Based Semantic Annotation
CBOW:	Continuous Bag of Words
CU:	Conceptual Units
CMTL:	Categorization Model Trainer and Loader
DL:	Description Logic
HAC:	Hierarchical Agglomerative Clustering
HMM:	Hidden Markov Model
IDF:	Inverse Document Frequency
IE:	Information Extraction
ILP:	Inductive Learning Processing
IoT:	Internet of Things
ISO:	International Organization for Standards
KB:	Knowledge Base
KIM:	Knowledge and Information Management System
KIMO:	KIM Ontology
LDA:	Latent Dirichlet Allocation
Lemon:	Lexicon Model for Ontologies
LSA:	Latent Semantic Analysis
LSTM:	Long Short Term Memory Network
ML:	Machine Learning
MUC:	Message Understanding Conference
MWE:	Multi Word Expression
NE:	Named Entities
NER:	Named Entity Recognition
NLP:	Natural Language Processing
OBIE:	Ontology Based Information Extraction

OG:	Ontology Graph
OL:	Ontology learning
ONTEA:	Ontology Based Text Annotation
OWL:	Web Ontology Language
POS:	Part of Speech
RDF:	Resource Description Framework
RDFS:	Resource Description Framework Schema
RE:	Relation Extraction
RL:	Reinforcement Learning
RNN:	Recurrent Neural Network
SAS:	Structure Analyzer and Segmenter
SCORE:	Semantic Content Organization and Retrieval Engine
SemAF:	Semantic Annotation Framework
SGM:	Semantic Graph Model
SOV:	Subject Object Verb
SVM:	Support Vector Machine
SVO:	Subject Verb Object
SW:	Semantic Web
TBD:	Taxonomy Based Disambiguation
TFIDF:	Term Frequency Inverse Document Frequency
VSM:	Vector Space Model
WaC:	Web as Corpus
WOE:	Wikipedia-based Open Extractor
WSD:	Word Sense Disambiguation
WWW:	World Wide Web
W3C:	World Wide Web Consortium

Chapter 1

Introduction

1.1 Background

An information distributed in the (World Wide Web) WWW is increasing from day to day due to the web that allows users to read and write information from any computers connected to the Internet [1]. A statistic reported in 2014 by the International Data Corporation[1] asserts that the digital universe is doubling every two years and will reach a size of 40 zettabytes[2] by 2020, from 4.4 zettabytes in 2013 [2]. Later on, semantic web (SW) constitutes an initiative to extend the web with machine readable contents and automated services beyond its current capabilities.

An important pre-condition to realize this goal of SW as an extension of the current web is the ability to annotate web resources with semantic information. In this demand, ontology plays a major role in supporting information exchange by extending the current syntactic interoperability of the web into semantic interoperability in a way that improves web documents accessibility and processability. Nowadays, WWW has become a source of information, where information is provided by humans for humans that need to be shared by human beings and software agents [3].

According to Browarnik and Maimon [4], humans gather information from text at the sentence or clause level, and not at the document (or corpus) level. Thus, extracting the semantic payload of the text would ideally include semantic labeling of the document using semantic analysis at the term, sentence or paragraph level. In line with this, ontology learning (OL) from web documents also aims to obtain domain knowledge covered by web documents by applying natural language analysis and statistical techniques in order to describe application relevant part of the world in a machine understandable way.

Originally in philosophy, ontology is concerned with the creation of a systematic way of describing existence [5], for instance, by classifying all entities in the universe into a hierarchy

[1] International Data Corporation (IDC) is a subsidiary of International Data Group (IDG), the world's leading technology media, research, and events company, https://www.idc.com

[2] zettabytes is one trillion gigabytes

of fundamental categories. In computer science, an ontology is considered as a formal knowledge representation [6] that describes all of the relevant things in a domain and the ways in which these things are related to each other. In this context, the modeled world knowledge, is used as a knowledge base for automatic semantic annotation and other SW technologies. In addition to this, integration of OL and semantic annotation is also beneficiary for both semantic annotations (by adding an option of considering evolved domain concepts) and ontology learning (by adding an option of self-learning or concept population). The manual construction of this knowledge base relies on ontology engineers assisted by one or more domain experts. This process can be complex, time-consuming and expensive depending on the size and complexity of the domain being modeled [4, 5, 7, 8].

Semantic annotation is going beyond simple textual annotations [8] as a process of tying semantic descriptions [9] for web documents by associating entities in a web document to their semantic descriptions. The semantic annotation process will pass through text analysis, concept extraction, ontology matching, and annotation phase in order to add explicit, formal, and unambiguous metadata to web documents. The main requirement of semantic annotation includes standard format, collaborative design, ontology support, heterogeneity support, document evolution, annotation storage and automation [10].

Several studies [1, 6, 9, 10] have indicated that ontology generation and semantic annotation with manual means are an expensive process and often does not consider multiple perspectives of document tagging and knowledge modeling. Therefore, the automation of ontology learning and annotation process is essential to provide the scalability needed to annotate existing web documents and reduce the burden of annotating new web documents. Reeve and Han [11] stated that automatic approaches for metadata acquisition promise scalability, and without these automations, SW will remain mostly a vision for a long time.

Automatic semantic annotation of web document is an open problem, but it is crucial to the realization of the SW [11], which requires the widespread availability of semantic annotations for existing and newly added web documents. Semantic annotation formally identifies concepts and relations between concepts in a document, and it is intended primarily for use by machines. For example, a semantic annotation might relate "*Addis Ababa*" in a text to an

ontology which identifies as the abstract concept "*City*" and links it to the instance "*Ethiopia*" of the abstract concept "*Country*", thus it removes ambiguity about "*Addis Ababa*".

Therefore, full implementation of SW requires widespread availability of semantic annotations for web documents in which manual annotation is working in different domains and languages; however, it has a knowledge acquisition bottleneck. Accordingly, the main target of this proposed thesis work is to present generic automatic approaches for semantic information extraction from web documents; for the purpose of ontology generation and semantic annotation that serve as a bridge between current web of links and the upcoming web of meanings.

1.2 Motivation

The growth of the WWW has increased the necessity to get solutions that can intelligently manipulate web documents and constitute valuable knowledge of a particular domain, but due to the unstructured nature of these documents, this knowledge cannot be exploited efficiently both by human beings and software agents (Figure 1.1 shows the report by Incapsula[3] on web traffic users analysis from 2012-2016 and in 2016, 48.2% of web traffic users are humans and the rest 51.8% are bad and good bots). Consequently, accessing web documents in correspondence with the meaning pertained to a user, constitutes the core challenge commonly referred to as "*semantic-gap*" [12] that limits the usability of these documents. Therefore, due to the explosion of information on the web and wide use of search engines for the desired information, the role of semantic annotation and ontology is becoming more interesting and significant for the current web.

The key idea in the SW, as articulated by Berners-Lee *et al.*[13], is to have data on the web defined and linked in such a way that its meaning is explicitly interpretable by software agents rather than just being interpretable by humans. Nowadays, as reported by Slimani [14], most of the documents on the web are not semantically annotated, which is difficult to realize SW vision that enable users to access web resources in terms of their meaning. Thus, this beneficial vision of SW motivates us to study on how to associate web documents with semantic information and allow software agents to perform intelligent tasks on behalf of human beings.

[3] www.incapsula.com

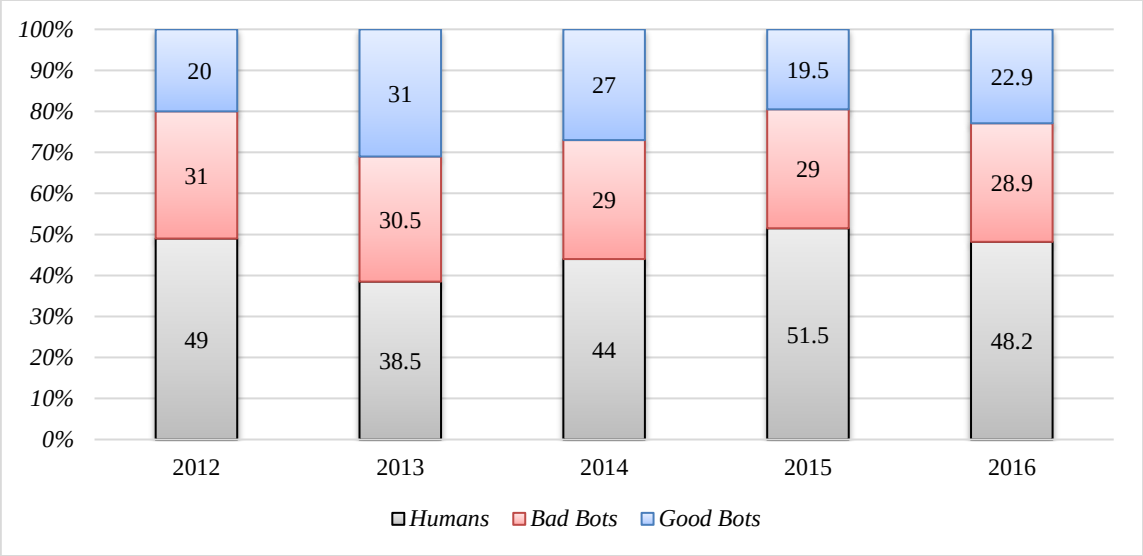

Figure 1.1: *Web Traffic Analysis Report by Incapsula*

1.3 Statement of the Problem

The exponential growth of the amount of data on the WWW requires automation of processes like searching, retrieving, and maintaining information. One of the difficulties that prevents complete automation is the fact that contents at the WWW are presented mainly in natural language, whose ambiguities are hard to be processed by software agents. Moreover, Internet users are also experiencing overwhelming quantities of online information and manual analysis of this data becomes nearly impossible. Thus, data analysis and information management would be performed using intelligent information management techniques to fulfill users' information requests by transforming existing data into a format that is machine processable and analyzable.

Traditional information retrieval systems mainly focus on text-based retrieval techniques, and they are usually based on keyword matching between user queries and indexed web documents. A problem of this text-based retrieval system is that the user might not enter enough and explicit terms in their query, since they may have no complete knowledge of the domain. As a result, information retrieval systems usually cannot provide appropriate and semantically related results. Thus, semantic tagging of existing web resource is a "*bridge*" to semantics based searching of the retrieval system.

Manual ontology generation and semantic annotation are expensive, tedious, error-prone, biased towards developers and inflexible tasks which lead to the knowledge acquisition bottleneck that creates difficulties in capturing the knowledge required by knowledge based systems and semantic web technologies. Although knowledge acquisition bottleneck is not a new problem in knowledge engineering [2, 3, 12], semantic web introduces new aspect to the problem with respect to the generic way of learning knowledge and semantic tagging. One way to tackle this challenge is the automation of ontology generation and document semantic annotation from web resource (ontology learning) to web resource (semantic annotation).

Nowadays, there is a good progress for linguistic annotation (the backbone of many supervised natural language processing (NLP) scenarios); however, semantic annotation that has great importance to facilitate semantic based web resource usage is still lagging behind. Even though the web data are being captured, stored and shared at an unprecedented scale, the semantic web technologies that helps people searching and usage of these media are lagging behind, due to their dependency on the success of other technologies including semantic annotation and ontology learning.

Semantic annotation is better if it involves, among others, word sense disambiguation, discourse structure, eventualities and their internal structures, semantic role labeling, temporal relations, detection of textual entailment, spatial relations and other deep NLP methods and processes. However, current automatic ontology learning and semantic annotation methods mainly rely on shallower NLP and statistical methods as discussed in [15], and they fail to handle deep semantic phenomena for knowledge extraction and handling. Accordingly, even if the implementation of deeper language specific NLP methods are required in order to abstract the text into a more semantically meaningful representation, deep language specific NLP is also expensive for morphologically rich languages.

To the best of the researchers' knowledge, there are gaps in existing semantic annotation works [16, 17, 18, 19] in terms of porting different preprocessing and postprocessing tasks supplementing the annotation process, different granularity level tagging, ontology learner integration, content to concept mapping, and document to annotation consistency management. Thus, generic semantic annotation framework that considers existing gaps is highly in need for the success of semantic web and related technologies.

1.4 Objectives

General Objective

The general objective of this research work is to propose and develop generic automatic semantic annotation framework for web documents.

Specific Objectives

In line with the general objective of this study, this research work is specifically aimed to meet the following specific objectives.

- assess different techniques and approaches employed so far in the web document analysis, ontology learning, and semantic annotation;
- understand behavior of natural languages and their commonalities;
- define the requirement of automatic semantic annotation;
- select best, appropriate, and reasonable approaches for semantic annotation that fits to the defined requirements;
- develop a comprehensive semantic annotation architecture;
- prepare corpus for ontology learning and semantic annotation;
- develop a prototype that demonstrates the proposed solution; and
- test and analyze the capability of the system.

1.5 Methods

Under this section, different methods of literature review, corpus preparation, prototype development and evaluation are discussed. These techniques will be used to identify concepts that facilitate solutions to semantic annotation and ontology learning related problems discussed under Section 1.3, and then evaluate the solutions through development of prototype systems.

Literature Review: The review of existing literature will help to analyze and understand the existing problems and also justify the value of the proposed solution. Thus, different literatures have to be reviewed that will include an investigation and analysis of information extraction, semantic interoperability, ontology learning, semantic annotation, and knowledge management. Exploratory and focused literature review methods will be used in a way that help us to explore existing problems and justify the solution.

Corpus Preparation: Corpus preparation methods are largely based on sampling methods from the social sciences. As corpus sampling technique systematic sampling and random sampling will be used to sample corpus that will be used for different models trainings and evaluations. Moreover, web as corpus (WaC) [20] is the recent advancement in corpus development technology. Following WaC principle, Wikipedia and Google web resource will be used as a corpus in the context of this research. Thus, in order to train relevant models related to ontology learning and semantic annotation, three sets of Amharic corpora consisting of relevant information will be gathered from Walta[4] information and public relations center, Google, and online Amharic encyclopedia, Wikipedia.

Development: In order to experiment the proposed solution, development of a prototype for ontology learning and semantic annotation will be conducted under this phase by solving one or more instances of the problem. Python programming language will be used to develop the prototype. In addition to this, Protégé, Keras, and Owlready will be used for semantic annotation and ontology learning prototype development.

Evaluation: The proposed research work will be evaluated using relevant metrics and analysis techniques such as precision, recall, and f-measure that properly measure the proposed automatic semantic annotator and ontology learner with the help of experts to observe and measure how well the proposed work supports a solution to the identified problems.

1.6 Scope and Limitations

The scope of this proposed research work is to design a framework for automatic ontology based semantic annotation of unstructured web documents excluding the following issues:

- A machine translation for the case of web documents containing multiple natural languages;
- Optical character recognition for scanned web documents; and
- Non textual document contents such as tables and figures.

[4] www.waltainfo.com: Walta is a private news and information service established in 1994 G.C. based in Addis Ababa, Ethiopia.

1.7 Application of Results

The semantic tagging and ontology learning end result can be applied in three general areas named as: information retrieval, computational linguistics, and artificial intelligence, which all aspire to equip computers with human knowledge and document understanding. In information retrieval, semantic tagging and ontologies are needed to organize and provide access to the ever-growing trove of digitized information; in computational linguistics, it drives the understanding and generation of human language; and in artificial intelligence, it supports efforts to make computers perform tasks that one would normally assume to require human expertise.

1.8 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 presents literature related to semantic annotation and ontology learning as a foundation of this research work. Chapter 3 presents related research works in the area of semantic annotation using pattern based, machine learning, and multi-strategy approach. The proposed framework is presented in Chapter 4 by discussing each components of the framework and summarizing the overall semantic annotation workflow. The experimentation of the proposed solution is presented in Chapter 5 with evaluation of the semantic annotator and ontology learner. Finally, Chapter 6 concludes the main contributions of this study and suggests an outline for the future works.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents review of essential literatures and analysis of current theories and methodologies which serve as a foundation for semantic annotation and ontology learning. From reviewed areas, natural language processing, information extraction, and semantic understanding are dependent on each other and also crucial for both semantic annotation and ontology learning. Thus, review of related fields including semantic web, annotation, ontology learning, information extraction, semantic understanding, and document processing are presented from Section 2.2 to 2.9 of this chapter.

2.2 Natural Language Processing

A language is the medium of human communication, and giving machines the ability to learn and understand human language is very essential for different real world applications. As a field, NLP is a subfield of computational linguistics that is concerned with application of linguistic methods for analysis and generation of natural language. Nowadays, NLP is a rapidly evolving research area that deals with comprehension and analysis of human-produced texts using computational methods as stated by Tixier *et al.* in [21]. A large amount of useful web resource is available in unstructured formats that includes reports, scientific papers, reviews, product advertisements, news, and emails. Thus, research in semantic annotation has focused on finding relevant information from these documents using NLP techniques such as sentence boundary detection, morphological analysis, and named entity recognition, which are employed as basic phases to annotate web documents with semantic information.

In the research area of NLP, there have been significant advances in machine learning approaches for automatic analysis of corpora, covering a range of linguistic levels, including: tokenization, POS tagging, parsing, semantic analysis, and discourse analysis [22]. In the literature, the following linguistic processing techniques have been discussed as NLP techniques for semantic annotation and ontology learning.

Sentence Splitting: Sequences of tokens are then (or simultaneous) divided into sentences using sentence delimiters; this is called sentence splitting, as a result corpus and document are transformed into a sequence of sentences.

Tokenization: Tokenization is the process of breaking up the sequence of characters in a text by locating the word boundaries, the points where one word ends and another begins [22] and it can be used in the initial phase of ontology learning and semantic annotation in order to extract a sequence of tokens from a given corpus and document.

Lemmatization: This is a normalization technique, used to map morphological variants to their corresponding base form or lemma. For example, the word "*mice*" becomes "*mouse*" or the word "*travelling*" becomes "*travel*". This is very important in information extraction and text mining, because "*travelling*" and "*travel*" would be considered as two different words without lemmatization. This process is often achieved by looking up a given word in a dictionary or a lemma list. Lemmatization is in contrast to the related process stemming, where a word stem is found by stripping off affixes, usually based on rules. As a process, lemmatization is usually preferred in favor of stemming, because it is often difficult to derive the original word from a stem (e.g., "*manager*" and "*management*", which both are reduced to the stem "*manag*").

POS tagging: In POS-tagging, each word must be assigned with its correct POS, such as noun, verb, adjective, or adverb; furthermore, most POS-taggers also give additional grammatical features, such as singular/plural number, tense, and gender. The number of tags used by different systems varies a lot, some systems use less than 20 tags [22]. The popular machine learning approach to POS-tagging is to use transformation based learning to learn local tag-combination constraint rules, which are used to eliminate candidate tags incompatible with the immediate context, or to select a tag that is required in a specific context.

Parsing: To process and understand natural languages, the linguistic structures of texts are required to be organized at different levels. Parsing used to understand how words are put together to form correct phrases or sentence along with the structural roles of words, and it plays a significant role in many NLP applications as it helps to reduce the overall structural complexity of sentences.

Word Sense Disambiguation (WSD): Words in a natural language are polysemy, having multiple senses and their semantic ambiguity can be resolved by other words in the context, since many words have associated multiple meanings (e.g., *mouse* can be used to represent either an animal or a computer device). Thus disambiguation of word senses helps in finding the correct context of such words.

Phrase Detection: Semantic ambiguity at the word level can be resolved by WSD, but if the ambiguity of a word can occur as part of the phrase it can be resolved by linguistic phrase detection.

Discourse Analysis: POS tagging, and parsing takes place at the level of word and sentence, each sentence in a text can be analyzed independently. However, language in real use exhibits structure beyond sentence boundaries. Pronouns are frequently used in discourse, detecting the entities to which those pronouns refer greatly increases the number of references to entities, which in turn, may greatly increase the number of extracted relations among those entities [23].

Higher Level NLP: As suggested by Liddy [24], in addition to the above linguistic processes higher level NLP also has a role in improving the semantic annotation process by providing an extra meaning that can amplify the current annotation with contextual and intentional knowledge.

In the field of textual semantic extraction, an important step forward has been realized through the availability of automatic NLP tools of listed processes under this section. These tools are generally based on linguistic methods such as morpho-syntactic pattern matching or on statistical methods such as frequency of term co-occurrences. However, language understanding and adaptation of the existing tools is challenging due to the following three basic factors in natural languages.

- Ambiguity, which is a critical challenge in which sentences and words may have different possible interpretations;
- Productivity of the language, since new word and language structure is encountered from time to time; and
- Cultural specificity of the language.

2.2.1 Feature Representation

Linguistic feature representation is very crucial for the performance of different real world applications related to NLP. The two major categories of text feature representation are one-hot and feature-embedding [25]. If there are few distinct features in the category, and no correlations between different features, the one-hot representation might be used. However, if there are correlations between different features in the group and gain some statistical strength by sharing the parameters, feature embedding is working better.

The use of the word-vectors[5] of feature embedding has great success in various NLP tasks, including named entity recognition (NER), POS tagging, and dependency parsing. In feature embedding each word is assigned a dense, low-dimensional real-valued vector, also called an embedding. Since word vectors are low-dimensional and also capture relations between words, the use of the word vectors have become more popular and successful than the traditional one-hot representations.

2.2.2 Word Maps and Language Models

For a machine to understand a natural language, first it is a must to develop maps of words with their meanings and interactions. It needs to build a dictionary of words, and understand where they stand semantically and contextually, compared to other words in the dictionary. To achieve this, each word has to be mapped to a set of numbers in a multi-dimensional space, in which similar words are close to each other, and dissimilar words are far apart in the vector space. As a word mapping, word embeddings [25, 26, 27] are successful unsupervised learning methods, because they do not require pricey annotation, rather they can be derived from already available unannotated corpora, which is probably their main benefit.

Language detection is required to classify textual document into the belonging written natural language and used as a primary step for some larger processes in NLP. Its automation is also possible, since natural languages are non-random, and they have regularities in the use of alphabetic sequences. Language detection based on alphabet and alphabet sequences stability is not new and has been proven for different languages. As a general paradigm for automated language detection, language models have to be modeled during a training phase and the

[5] A recent big idea in natural language processing is that "*meanings are vectors*" in which vectors that are close to each other represent similar meanings.

comparison of the textual document has to be computed against the model during language identification. In this research work, the main purpose of the detection is to optimize the processing time of semantic annotation by considering only documents that has linguistic related resources and generify the framework for other natural languages adaptability.

As a language detection approach, character-based n-gram assumes that language structure follows certain alphabetic sequences and should not need a large set of training dataset because, alphabetic sequences in the language is sufficiently prevalent even in a smaller set of text. However, the existing challenge of using this approach is the closer any two languages, the harder to distinguish them. The language closeness may result from common historical roots, borrowing, or overlapping character-to-integer mappings inside the computer. Therefore, the reliability of any language model will be dependent on the size and internal consistency of the corpus from which the model is derived.

2.3 Information Extraction

Information extraction (IE) refers to the automatic extraction of structured information such as entities, relationships between entities, and attributes describing entities from unstructured sources. The extraction of such structured information from noisy, unstructured sources is a challenging task engaged in various research communities including NLP, machine-learning, information retrieval, database, and document analysis. In the context of this study, both semantic annotation and ontology learning require different IE techniques to extract information from a given document and corpus respectively.

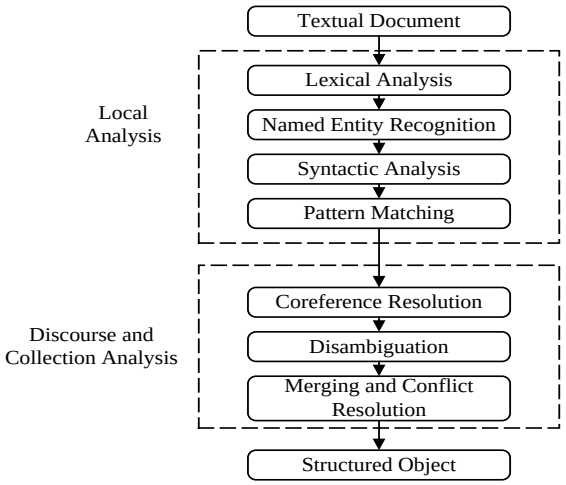

Figure 2.1: *Typical stages in the Information Extraction Process*

As technique, IE often is applied to the automation of semantic annotation of textual resources and incorporates different components as shown in Figure 2.1 [28]. Most of the technologies used for IE was developed in response to a series of evaluations and associated conferences called Message Understanding Conference (MUC), held between 1987 and 1998 [29].

2.3.1 Structure Analysis

Document layout is ignored and only the text itself is processed, usually as linear sequences or even bags of words in most of existing information extractions for semantic annotation. This allows the information extraction to be simpler and cleaner, at the cost of a challenges on valuable information extraction. Thus, structure analysis and segmentation analyzes a given document and creates a structured representation in a way that improves the document information extraction quality by focusing on most essential part of the document, since it is by far easier and better to obtain good results on organized data than with a smart algorithm over unorganized data.

Table 2.1: *Document Metadata and Structure Template Elements*

Metadata/Structure	***Metadata/Structure Elements***
Metadata	Title, Author, Created Date, Modified Date
Structure	Abstract**,** Introduction, Conclusion, Paragraphs, Sentences, words, non-textual contents, and References

The structurally segmented representation is mainly used to reduce complexity of a document and make it easier for processing and understanding. The document has to be transformed from the full text version into structural elements version (example of basic structural elements are shown in Table 2.1). Up on this, dimensionality reduction allows an efficient document manipulation and enhances the performance of document processing algorithms both in speed and accuracy. Basically, one of the major advantages of electronic document is an explicit structure that can be partitioned into a hierarchy of physical[6] and logical[7]

[6] Hierarchy of physical components, such as pages, columns, paragraphs, text lines, words, tables, and figures.

[7] Hierarchy of logical components, such as titles, authors, abstracts, and sections.

components or both. This structural information can be very useful in retrieving information contained within the document specifically for annotation purpose in the case of this research.

In line with this, abstract, and conclusion sections of a document express essential information in a compact and often repetitive manner, which makes them an optimal section to understand the document. This also led us to mainly focus on this section to extract and map document contents to the ontological concepts for understandability and process-ability of the document with software agents.

2.3.2 Information Extraction Approaches used in Semantic Annotation

Applying information extraction on text is related to the problem of text simplification to create a structured view of the information that exists in a free text. Nowadays, different techniques can be used to extract information, from simple pattern matching to complete processing methods based on symbolic information and rules or statistical methods and machine learning. The semantic annotation research works use several methods of information extraction from Web documents such as rules, wrappers, and patterns [30]. The significance of information extraction is determined by the growing amount of information available in unstructured form (i.e., without metadata), for instance on the Internet.

a. Rule based Information Extraction

Rules are typically handcrafted and define how entities can be found in unstructured documents (examples of systems using rule based information extraction are AeroDAML and KIM). As stated by Reeve [30], rule based systems require an expensive maintenance of pre-defined rules when the data source or problem domain is changed. In order to reduce the effort required to build and maintain manual rules for the semantic annotation most of the earliest systems use inductive machine learning algorithms for automatically learning rules**.** Inductive learning processing (ILP) uses a training corpus to find common patterns of objects to be extracted through inductive machine learning processes. SemTag [18] platform is an example of the system using inductive machine learning for semantic annotation.

In information extraction related literatures [28, 29], combining ILP methods with NLP methods is good for both data extraction and semantic annotation tasks. The two reasons of the combination are: first, many data extraction/annotation patterns are represented in regular expressions in which NLP techniques make good use of those regular expressions, while ILP

techniques help to find and generate those regular expressions. Second, especially for semantic annotation tasks, ILP techniques can easily associate ontological definitions of objects with the learning patterns, which is hard for pure NLP techniques.

b. Wrapper based Information Extraction

Wrappers can be handcrafted, or they can be learned from examples prepared by experts manually. The manual wrapping requires the user to mark areas of interest within a document [30] and machine can then extract entities from documents with a similar structured format as of the manually marked-up document. Wrappers can also be linguistic-based, where the wrapper induction process discovers linguistic rules for identifying entities. MnM [31] and Ont-O-Mat [1] annotation tools are examples of the tools that use wrapper based information extraction for semantic annotation.

c. Pattern based Information Extraction

Pattern based information extraction can exploit known linguistic patterns in order to find entities, and this technique is widely used in semantic annotation methods. Armadillo semantic annotation tool [10] is an example of a system that uses pattern-based information extraction.

d. Advanced IE Techniques

Advancing information extraction techniques for semantic annotation will improve performance of the annotator and also increase accuracy of semantic annotation process. Several researches on IE have documented importance of using advanced NLP techniques and ontology. The following are advanced IE methods rely on advanced NLP, ontology, and Wikipedia as discussed in the literature.

Historically, most natural language processing systems have been designed to process unstructured text, which consists of natural language sentences. The meaning of unstructured text depends entirely on linguistic analysis and natural language understanding [29]. NLP components such as tokenizer, POS tagger, sentence splitter, NER, and WSD are used in information extraction processing. However, higher level of natural language processing such as pragmatic level processing has not been used for semantic annotation even if its existence will improve the quality of the annotation and used to understand the context of the document.

Ontology based information extraction (OBIE) has emerged recently as a subfield of IE. Here, ontologies are used by the information extractor, and the output is generally presented using ontology [32]. OBIE utilizes formal ontologies to guide the extraction process [33].

Wikipedia-based open extractor (WOE) is proposed by Wu and Weld [34] to generate relation-specific training examples of matching Wikipedia Infobox[8] attribute values to corresponding sentences in Wikipedia articles. This automatic construction of training examples by heuristically matching Wikipedia infobox values and corresponding text is used to generate an un-lexicalized, relation-independent (open) information extractor. As of April 2017, the Amharic version of Wikipedia[9] contained more than thirteen thousand articles which are expected to offer an overview of its subject at the beginning of the article and usually starts with a definition, a summary, or a short description of the subject. Often, a box next to the summary offers structured information about the article's subject in a tabular form and these so-called infoboxes containing facts about the described subject that is displayed as attribute-value pairs.

2.3.3 Data (Text) Mining

Data mining is a field of research that is concerned with deriving relevant information from large amounts of data. Most of data-mining researches assume that information being "*mined*" is already in the form of a relational database [35]. Unfortunately, for many applications, available electronic information is in the form of unstructured natural language documents rather than structured databases. Data mining is a hot topic in computer science and deals with the extraction of useful information from large volumes of data and it is an "*umbrella-term*" for a set of methods in computational world.

Text mining methods are the most common data mining methods in the field of ontology learning and text understanding. Text mining process is dedicated to discovering and extracting knowledge from unstructured textual data. It is a multidisciplinary field, involving information retrieval, text analysis, information extraction, clustering, categorization, visualization, database technology, machine learning, and data mining [35]. As reported by

[8] An infobox is a set of tuples summarizing the key attributes of the subject in a Wikipedia article.

[9] A complete copy of all Amharic Wikimedia wikis, https://dumps.wikimedia.org/amwiki/

Sailaja [36], text mining techniques become more complex as compared to other data mining methods due to the unstructured fuzzy nature of natural language text and majority of concerns in text mining are posed by the particularities of the natural language.

In text mining, semantic understanding is more crucial than statistical understanding particularly for tasks related to document understanding and annotation [37]. Thus, less relevant content removal, stemming, noisy data removal, word sense disambiguation, tagging, collocations (compound or technical terms), syntactic analysis, tokenization, text representation, and semantic analysis are basic processes for semantic based text mining.

2.3.4 Named Entity Recognition

NER is the process of identifying a word or a phrase that references a particular entity within a text and addresses the identification and classification of predefined types of named entities (NE), such as organizations, persons, locations, dates, and currencies. The automation of document annotation is typically implemented with the help of different IE techniques, among which NER is used to identify document contents to be mapped into ontological concepts. Entity recognition is an essential task used in document information extraction and tagging, since most of the data that need to be annotated are those of the named entities such as persons, locations, organizations, and currencies.

NER is a two-phase process named as identification and classification. Identification is concerned with marking the presence of a word/phrase as named entity within a given text while classification is concerned with denoting the predefined role for the identified named entity [23]. NER has been researched for several decades and early systems [38, 40] used handcrafted rule-based algorithms, while modern systems [39] usually use machine learning techniques and there are also some systems using hybrid techniques. Handcrafted rule based systems usually have good results, but they need a lot of effort by experienced linguists, whereas machine learning techniques use a collection of annotated documents to train the classifier for a given set of named entity classes. Different research attempts have been conducted for NER including [38, 39, 40]. The NER seems easy with a prepared database of all kinds of entities. However, it is quite challenging due to two reasons: entity databases are often incomplete; and the same word/phrase can refer to different entities (or none entity)

depending on the context, therefore, lexical form is insufficient to determine the named entity from the unstructured text.

2.3.5 Relation Extraction

Relation extraction (RE) is an extraction task that can be used in the process of information extraction and usually comes after entity recognition. Once all significant entities are identified, this task is used to connect together those entities that are collected from the document and to assign the correct label (relation name) to this connection. As studied by Byrne [41], methods of RE range from those using exclusively hand-crafted patterns and rules at one end of the range, to those relying entirely on probabilistic methods on the other range. In RE, most of the approaches rely on the mapping of syntactic dependencies, such as subject-object-verb (SOV), into semantic relations, using either pattern matching or other strategies, such as probabilistic parsing or clustering of semantically similar syntactic dependencies. Thus, machine learning based relation extraction approaches have been classified according to their degree of learning as supervised, semi supervised and unsupervised relation extraction approaches [23].

Semantic relations extracted from texts are useful for several applications, including question answering, information retrieval, semantic annotation, and construction and extension of lexical resources and ontologies. Thus, extracting semantic relations between entities in natural language text is a crucial step towards natural language understanding applications.

2.4 Semantic Understanding

The computation of semantic understanding is exploited in several research fields, including artificial intelligence, knowledge management, information retrieval, and other several fields. The challenge of semantic understanding is to find a method that can simulate the thinking process of human [42]. Relatively, large number of semantic similarity metrics have been proposed in the literature [43, 44] and there are mainly two categories of approaches in measuring semantic similarity for semantic understanding, namely knowledge-based and corpus-based approaches [45]. The knowledge-based uses structure of the semantic network between concepts in the knowledge graph (e.g., path length and depth), while, corpus based uses information content of a given word from large corpora such as Wikipedia.

As stated by Sailaja [36], a long tradition in computational linguistics has shown that contextual information provides a good approximation to word meaning understanding, since semantically similar words tend to have similar contextual distributions. In concrete, distributional semantic models [46, 47] use vectors that keep track of the contexts (e.g., co-occurring words) in which target terms appear in a large corpus as proxies for meaning representations, and apply geometric techniques to these vectors to measure the similarity and understand meaning of the corresponding words.

Distributional semantics is a successful, scalable, and flexible approach to meaning as reported by Gupta *et al.* in [48]. In distributional semantics, word embedding is a technique that treats words as vectors whose relative similarities correlate with semantic similarity and this technique is one of the most successful applications of unsupervised learning. NLP systems are traditionally encode words as strings, and word embedding[10] is an alternative technique in NLP whereby words or phrases from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary size, and the similarities between the vectors correlate with the words' semantic similarity.

Vector space models (VSMs) have been used in distributional semantics and the development of different models are used for estimating continuous representations of words. VSM is used in the areas of information extraction and machine learning [49] for document and query representation and modeling. It was initially designed as a model to represent arbitrary text documents as vectors from a common vector space. VSM uses frequencies in a corpus of text as a clue for discovering semantic information. Latent dirichlet allocation (LDA) [50] and latent semantic analysis (LSA) [51] have been proposed under distributional semantics.

Moreover, Google, Wikipedia and linguistic pattern are also used for semantic understanding of documents. In Google semantics, Google events have to be captured to extract background knowledge about the search terms available on the web (Google). The Google event *x*, consisting of the set of all web pages containing one or more occurrences of the search term *t*, thus embodies, in every possible sense, all direct context in which *t* occurs on the web. This constitutes the Google semantics of the term. For Wikipedia semantics**,** several research

[10] Word2Vec and Glove are different variants of word embedding.

efforts have been conducted for extracting explicit or implicit relations from Wikipedia to represent semantics of concepts or words [52]. Wikipedia based method operates in two steps. First, it extracts a set of sentences and other features, containing word information, and tag words with the extracted features. Second, it calculates semantic similarity between words based on several factors, such as a term frequency, probability distribution, and term co-occurrences. In pattern based semantics, Hearst's [119] seminal work in this area opened a line of research followed by many authors who have focused on the identification of specific relations like hypernym, meronymy, and antonym for understanding words/phrases semantics.

As indicated by Grainger *et al.* [53], if someone searches for the term "*server"* in information technology domain, it has a very different meaning (a computer server) than in the restaurant domain (a waiter/waitress), and if someone is using a job search engine, it could actually represent either meaning depending upon the user's context. With the dramatic increase in the amount of documents available in digital forms, it has become a necessity to categorize large texts (documents) into specific domain for efficient semantic understanding and retrieval of information and knowledge [54]. Nowadays, with the explosive growth of the web data, algorithms that can improve the categorization efficiency while maintaining accuracy, are highly desired in semantic understanding and processing.

Table 2.2: *Relationship Pairs in Word Embeddings from English word2vec Model*

Relationship	***Example 1***	***Example 2***	***Example 3***
France – Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big – bigger	small: larger	cold: colder	quick: quicker
Miami – Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein – scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy – France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper – Cu	Zink: Zn	gold: Au	uranium: plutonium
Berlusconi – Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft – Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft – Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan – sushi	Germany: bratwurst	France: tapas	USA: pizza

As semantic understanding methods from unstructured text, word embedding variants such as word2vec [26, 55], Glove [27] and FastText [56] are used as effective techniques to elicit knowledge from large text corpora in an unsupervised manner. As word embedding model, word2vec takes unlabeled corpus as input and generates word vectors as output for each unique words within a corpus. It is widely featured as a member of the "*new-wave*" for machine learning algorithms based on neural networks, commonly referred to as deep learning (though word2vec itself is rather shallow). Table 2.2 shows list of relationship pairs [55] as an example using word embeddings (word2vec) from unlabeled English Wikipedia corpus. Word2vec is using large amounts of unannotated plain text to learn relationships between words automatically.

Word2vec model cluster each word according to their contexts by assigning similar words into closest coordinates and uses two different architectures named as continuous bag-of-words (CBOW) and skip-gram to produce the distributed representation of words [57]. CBOW predict word by looking at given context words in a certain window size, whereas skip-gram uses the current word to predict surrounding context. Even if it is said that CBOW is faster than Skip-gram model, skip-gram is better for infrequent words [58]. Therefore, to achieve the semantic web goal, software agents must be equipped with text semantic understanding of web documents and publish their understanding as an annotations in a form accessible to other software agents, using semantic web languages such as RDF and OWL.

Learning context of words help us to learn good embeddings, let's see the following example of word2vec learning from two sentences, "*Lion lives in forest"* and *"Tiger lives in forest"* and if the model is trained with (input: *Lion* output: *Forest*) and (input: *Tiger*, output: *Forest*) this will eventually force the model to understand that, *Lion* and *Tiger* both are related to *Forest* thus placing *Lion* and *Tiger* closely in the embedding space. Once word2vec is modeled from unlabeled textual corpus, the model provides two basic usage named as distance usage and analogy usage. The distance usage provides lists of words closely related to a particular word from the vector model with cosine similarity of related words which indicates how words are closer to each other in a vector space.

2.5 Machine Learning

ML is a general term for the subfield of artificial intelligence in which the intelligence concerned with development of algorithms and techniques that allow computers to "*learn*" and perform different tasks. It focuses mainly on how to detect patterns in a data to predict future data, or perform other kinds of decision making under uncertainty. As presented by Harrington [59], during the last half of the twentieth century, majority of workforces in the developed world have been moved from manual labor to what is known as a knowledge work. Thus, to assist knowledge workers, machine learning (ML) techniques were widely used in computational world by learning patterns from train dataset to predict future dataset as shown in Figure 2.2.

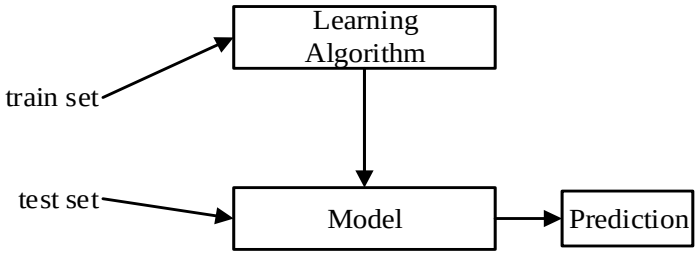

Figure 2.2: *Machine Learning Steps using Train and Test Dataset*

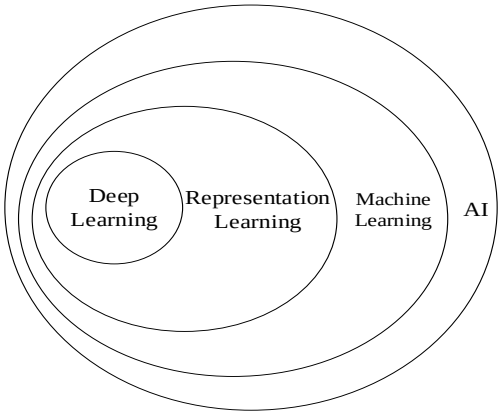

Figure 2.3: *A Venn diagram showing learning and AI dependency*

Modern NLP tools have used ML algorithms and statistical modeling to overcome the difficulties of web document content analysis. According to Tixier *et al.* [21], most widely used techniques in text analysis are clustering and classification algorithms. Even if, machine learning techniques are used widely in the field of computer science, most of the classical machine learning algorithms are data-driven, so the performance of these methods depends

on the quality and quantity of the data used for the training. Typical processes in machine learning are illustrated in Figure 2.2 and the ML techniques are classified as supervised, semi-supervised and reinforcement machine learning techniques in terms of the supervision used for learning techniques and interaction with the environment. Learning (i.e., deep learning, representation learning, and machine learning) to artificial intelligence (AI) dependency is illustrated in Figure 2.3 [60] as Venn diagram.

2.5.1 Supervised Learning

Supervised learning, aims at predicting "*outputs*" of the target function based on known training "*input-output pairs*" from the function. This learning techniques suffer from a fundamental data bottleneck problem, since they require huge data annotated for a specific task to be learned. The process of annotation in turn requires human time and is thereby inherently connected to high costs, both in terms of time and money. As a result, language specific resources and features based learning is expensive to adapt existing supervised ML models for new languages and domains. There are some common algorithms that lie under supervised learning such as linear regression, SVM, and nearest neighbor.

2.5.2 Unsupervised Learning

In unsupervised learning, there is no supervision and only input data is given with aims to find regularities from input dataset. There is a structure to the input space such that certain patterns occur more often than the others, and the machine want to see what generally happens and what does not. The most common type of unsupervised learning method is clustering which comes in a large variety of forms. The general idea is to assign entities from input space X into clusters without a predefined criterion or set of examples for cluster membership.

Semi-supervised learning is halfway between supervised and unsupervised learning. In addition to unlabeled data, the algorithm is provided with some supervision information, but not necessarily for all examples. Semi-supervised learning is more useful whenever there are far more unlabeled data than that of labeled data. This is likely to occur if obtaining data points is cheap, but obtaining the labels costs a lot of time, effort, or money and it is the case in many application areas of machine learning (e.g., speech recognition that requires listening and labeling, web pages classification that requires reading to label each web page into different classes).

2.5.3 Reinforcement Learning

Reinforcement learning (RL) is a computational approach to understanding and automating goal-directed learning and decision-making [61, 62]. RL learns the behavior through trial-and-error interactions with a dynamic environment and it uses a formal framework defining the interaction between a learning agent and its environment in terms of states, actions, and rewards. The algorithms (e.g., q-learning, deep adversarial networks, and temporal difference) under this category learns in an iterative fashion and utilizes the observations collected from the interaction and take actions to minimize risk and maximize benefits. Real time decision, robot navigation, learning tasks, and skill acquisition are some of possible RL application areas. Table 2.3 shows ML categories summary with their examples and application areas as an example.

Table 2.3: *Summary of ML Categories*

S.No.	***Machine Learning Categories***	***Algorithm(s)***	***Application(s)***
1.	Supervised	Classification	Domain classification, Fraud detection
		Regression	Digital data growth prediction
2.	Unsupervised	Clustering	Hierarchical agglomerative clustering
		Dimensionality Reduction	Structure discovery, Big data visualization
3.	Reinforcement	Q-Learning	Game AI, Skill acquisition

2.6 Neural Networks

The human brain consists of an estimated 100 billion nerve cells or neurons [63]. Each neuron typically receives many thousands of connections from other neurons constantly receiving a multitude of incoming signals, which eventually reach the cell of the body. Here, they are integrated together in some way and if the resulting signal exceeds certain threshold, the neuron will generate a voltage impulse in response. This is then transmitted to other neurons via a branching known as the axon.

Artificial neural networks (ANN) understood as simplified models of the networks of neurons that occur naturally in the human brain. Learning with neural networks was proposed in the

mid-20th century and it yields an effective learning paradigm and has recently been shown to achieve cutting edge performance on several learning tasks. ANNs are often used for statistical analysis and data modeling, in which their role is perceived as an alternative to standard nonlinear regression or cluster analysis techniques. ANN[11] has several advantages for NLP and semantic understanding including, fit to any kind of data, hidden layers that can be used as word look up tables, and dense distributed word vectors are used for word representation in an unsupervised fashion, which is used for different NLP systems.

Humans do not start their thinking from scratch every second, as we read a written document, we understand each word based on our understanding of previous words. We do not throw everything away and start thinking from the scratch. However, traditional ANNs cannot do this, and it seems like a major shortcoming. Thus, recurrent neural network (RNN) was proposed to address this issue as a kind of neural networks with loop that allows information persistency.

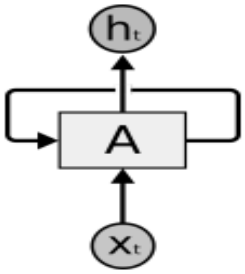

Figure 2.4: *Diagram of Recurrent Neural Networks*

As it can be seen from Figure 2.4, a chunk of neural network, *A*, looks at some input x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next. In theory, RNNs are absolutely capable of handling "*long-term-dependencies*". However, in practice, RNNs do not seem to be able to learn long term dependencies as presented by Bengio *et al.* in [64]. As a solution, Senior [65] presented long short term memory network (LSTM) as a special kind of RNN that has capability of learning long-term dependencies.

[11] Machine learning techniques, inspired by the architecture of the human brain and made possible by recent advances in computing power, have been making waves via breakthrough results in image, speech, and natural language processing.

Deep Learning [60, 66] is another name for a set of algorithms that use a neural network as an architecture. Even though neural networks have a long history, deep learning became more successful in recent years due to the availability of inexpensive, parallel hardware (GPUs, computer clusters) and massive amounts of data.

2.7 Semantic Web

Syntactic web performs information retrieval simply by finding matches of words or phrases that we indicate in the query for the search engine. The semantic web does not have the same nature with existing web, but rather it is an extension of syntactic web by providing semantic support to the content of the web resources in a way that allows both humans and software agents to work together using information from the web as stated by Berners-Lee *et al.* [13]. The semantic web has to go beyond simple bag-of-words approach currently used by search engines and get closer to the meaning of the texts. The most important thing missing in syntactic web is a linkage between web document texts and external resources encoding semantic information.

SW is a well-known technology that requires semantically tagged web documents to facilitate machine-to-machine communication. Semantic annotation is used to fulfill this requirement and named as semantic web dream enabler. As a term, semantic web was first introduced in Tim Berners-Lee's 1999 book, "*Weaving the Web*", and 1999 is marked as the birth year of the semantic web. After 2001, the idea of the semantic web has become more and more accepted by academic researchers, industrial inventors, and even many other people without any computer science background with common purpose of making the web machine understandable to software agents.

Tim Berners-Lee, the inventor of WWW, describes SW as a component of web 3.0[12] that representing a major intelligent addition to the web [13]. A core goal of SW technology is to bring progressively more meanings to existing web, and acceptable method of doing this is by annotating documents with different kinds of metadata and help document owners to easily organize their documents for better search facilities. Thus, web users can search for web resources not only using keywords, but also using well-defined general concepts that

[12] Third generation Web, is a web where all information is categorized and stored in such a way that a computer can understand it as well as a human.

describe the domain of their information need. To do this, manual annotation is time-consuming and error-prone due to the fact that semantic annotations must be made in a formal language, unlike natural languages, to achieve its intended goal [67].

On the demand side of the semantic web, a core capability is improving the precision of web search with the help of semantic annotation that has to be unambiguous and sufficiently detailed to support the search engine in making fine-grained distinctions in understanding of web documents. Another core capability is to excel the level of document retrieval for user queries by generating pragmatically and stylistically appropriate responses for user queries. To attain these capabilities, Java *et al.* [67] have suggested that an intelligent agent must rely on very detailed semantic annotations of textual documents. In recent years, automated semantic annotation is being introduced into the research at an increasing rate with the objectives of enabling machines to "*understand*" the semantic meaning of web data [68]. Previous studies [69, 70] indicated that semantic annotation usually requires annotation schema or reference ontologies with explicitly-defined, consensually-agreed, and machine-understandable semantics to annotate the content. This helps both software agents as well as human beings to work cooperatively together and exchange knowledge and resources [71].

In general, there are three levels of knowledge encapsulation for the semantic web. The first level encapsulates syntactic information about knowledge using XML and RDF (S). The second level encapsulates semantic information about knowledge using ontologies, through ontology languages such as OWL. The third level uses reasoning and security techniques to provide ways of manipulating and protecting knowledge.

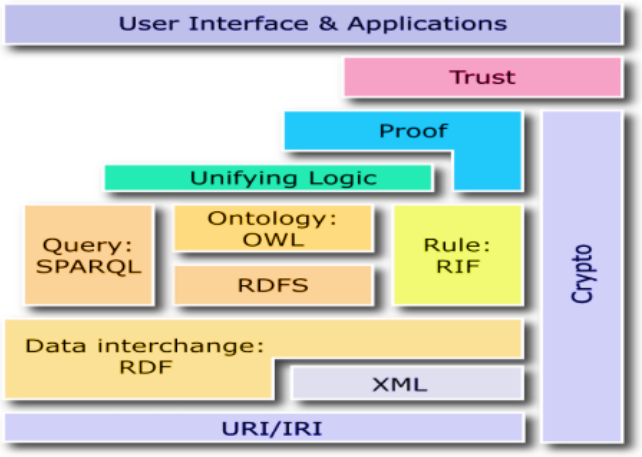

Figure 2.5: *Semantic Web Building Blocks*

Resource description framework (RDF) is a standardized model used for the purpose of data interchange and it is adopted by World Wide Web consortium (W3C) as one of the standard formalism of knowledge representation on the Web. It provides the semantic web application with interoperability feature, because RDF is readily available for any program and facilitates data merging, no matter what schema used. Storing knowledge using this standard is done by decomposing it into three triples. One triple is composed of resource, attribute and value in order to describe web resources. In another way, it is composed of a resource (object), named property (attribute) and value for the property (value) [1] and these triples can be expressed in three ways as tables, xml files, or graphs. The semantic web stack building block shown in Figure 2.5 [13, 72] is an illustration by Tim Berners-Lee, and depicts hierarchy of languages used to construct the semantic web and each layer exploits and uses the capabilities of the layers below.

Sánchez *et al.* [73] mentioned that SW relies on two basic components, ontologies and semantic annotations. On the one hand, ontologies are knowledge structures representing the semantics of domain concepts and their interrelations in a machine-readable way. On the other hand, annotations represent a specific sort of metadata that provides references between entities appearing in resources and domain concepts modeled in ontology.

2.8 Semantic Annotation

As reported in 2002 by Bontcheva *et al.* [74], up to 2012 more than 95% of human-to-computer information input will involve textual language. As of the report in 2012, taxonomic and hierarchical knowledge mapping and indexing will be prevalent in almost all information-rich applications. There is a tension here between the increasingly rich semantic models in information technology systems on the one hand, and the continuing prevalence of human language materials on the other. Thus, the process that tying semantic models and natural language together is referred to as semantic annotation as a way to present possible solution for this existing tension.

An important pre-condition for realizing SW goal is the ability to annotate web resources with semantic information. To realize this goal, in recent years, various researches have been conducted for semantic annotation after the advent of semantic web technology. As a result, semantically annotated data can be automatically integrated across different sources, and such

data are ready for inference of additional knowledge. As a process, semantic annotation is the process of labeling documents with the semantics of their contents as shown in Figure 2.6 and discussed in Chapter 1 with the purpose of enabling computers to understand human language so that they can perform tasks that are more intelligent.

Semantic annotations turns human-understandable content into a machine understandable form and serve as bridging technologies from syntactic web into semantic web. The semantic annotation mainly supports the advanced concept searching, information visualization using ontology, and reasoning about web resources. Thus, the main feature of semantic annotation is the conversion of existing syntactic structures into knowledge structures [75]. Semantic annotation formally identifies concepts and relations between concepts in documents, and is intended primarily for use by machines [76]. For automatic semantic annotation, two types of systems that learn how to annotate documents are used as supervised and unsupervised way of learning.

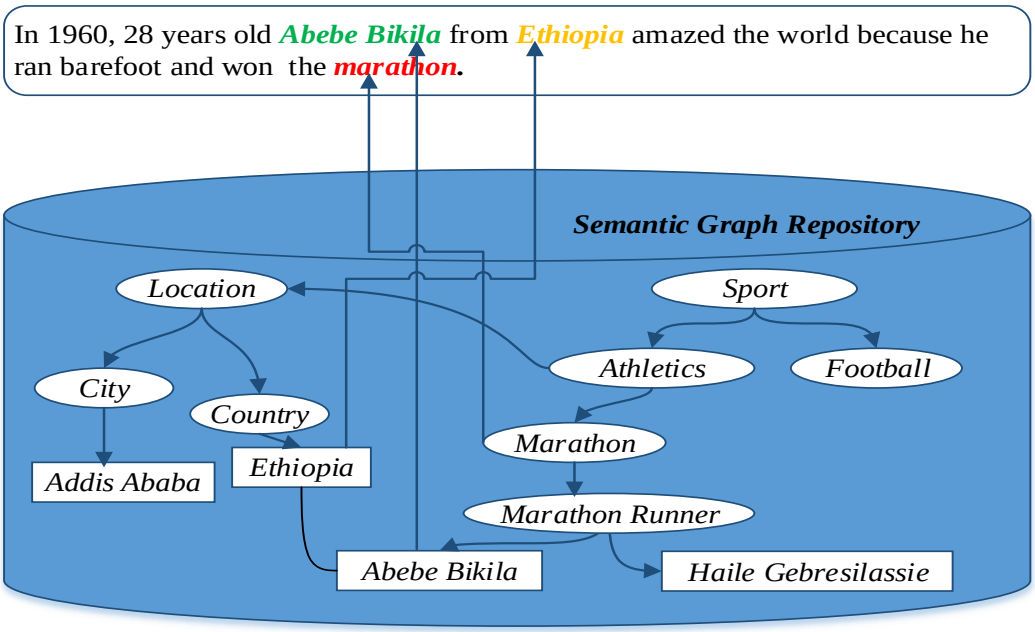

Figure 2.6: *Example for Content to Concept Mapping in Semantic Annotation*

Automatic semantic annotation is achieved by automating different algorithms used for annotation. It has also become increasingly important research topic, which enables many applications such as highlighting, indexing, retrieval, categorization, and information extraction possible for better digital resource information management. Over the recent years,

major search engines have been redesigned their search methods to accommodate semantic information to provide rich search results and semantic based question answering. They have also developed large-scale entity repositories, such as Google's knowledge graph, Yahoo's web of objects, and Bing's satori [77]. They are large-scale annotation technologies as road map into modern information retrieval system.

2.8.1 Semantic Annotation Standards

In Dublin, Ohio, March 1995, an agreement about the so-called Dublin core metadata element sets were conducted, which is named by the location of the workshop. This is a significant effort showing that humans and researchers want to have a universal (semantic) annotation standard to specify web documents. Bunt [78] described international standards organization (ISO) standard 24617-6, "Principles of Semantic Annotation", sets out the approach to semantic annotation that characterizes the ISO semantic annotation framework (SemAF). The purpose of ISO 24617-6 is to provide support for the establishment of a consistent and coherent set of international standards for semantic annotation in order to improve the interoperability of semantically annotated resources.

2.8.2 Semantic Annotation Requirements

Bowers *et al.* [79] studied and identified three potential requirements for effective semantic annotation as well defined ontologies, methods to annotate objects, and algorithms to make use of semantic annotation. In addition to this, as formulated by Uren *et al.* [10] there are seven requirements (listed here below i - vii) for semantic annotation systems.

i. **Standard format**: Using standard format is preferred, wherever possible, because the investment in marking up resources is considerable and standardization builds in future proofing. Particularly for annotation systems, standards can provide a bridging mechanism that allows heterogeneous resources to be accessed simultaneously and collaborating users and organizations to share annotations. Two types of standard are required [10], standards for describing ontologies such as the web ontology language OWL and standards for annotations such as the W3C's RDF annotation schema.

ii. **User-centered/collaborative design:** The system should provide easy to use interfaces that simplify the annotation process and facilitate collaboration between users.

iii. **Ontology support:** In addition to supporting appropriate ontology formats, annotation tools need to be able to support multiple ontologies.

iv. **Support of heterogeneous document formats:** Dealing with multiple document formats is also a prerequisite for integrating annotation into existing work practices.

v. **Document Evolution:** Documents and ontologies can be changed so that new or modified markup is required, which leads to document markup maintenance issues. The proposed approach for document evolution is stated as the separation of the semantic annotation into short and long term annotation in a way that helps for managing document evolution. Long term semantic annotation is the annotation of the document that might be affected rarely and consistent with the document for the long period of time. Short term semantic annotation is the feature of the document that will be changed frequently and affected by the change of the document.

vi. **Annotation storage**. There is no universally winning choice for storing the annotation content; either to be stored separately from the annotated resources (stand-off annotations), or to be embedded into resources (in-line annotations), but using stand-off annotation is the easiest way to manage the annotation separately from the web document.

vii. **Automation:** automatic markup of document collections is required to facilitate the annotation of large document collections economically and solve the knowledge acquisition bottleneck of manual semantic annotation.

2.8.3 Semantic Annotation Phases

As studied by Gao *et al.* [80], semantic annotation consists of two major phases: ontology-based lookup and reference disambiguation. The ontology-based lookup is concerned with identifying all candidate mentions of concepts and the reference disambiguation then uses contextual information from documents as well as knowledge from the ontology to disambiguate the mentions to the correct ontological concept. In line with this, Nagarajan [81] also classified semantic annotation process into three primary steps: entity identification, entity disambiguation and annotation. In related research effort, Große-Bölting *et al*. [82] have stated that the automatic semantic annotation consists of four processes named as: concept extraction, concept activation, annotation selection, and evaluation. More generally, semantic

annotation using ontology based IE can also broadly classified into three steps named as extraction, alignment, and annotation.

Even if annotation procedures presented in the literature are varied depending on the approaches and methods used by the researcher, the following are common and basic phases used as procedure for automatic semantic annotation.

i. **Preparation phase:** analyzes a document in order to normalize and convert it into an appropriate way for the content (i.e., concepts and instances) extraction.

ii. **Content extraction and categorization phase:** content extraction is used to extract candidate concepts representing feature of the target document while **c**ategorization is used to classify the document into their corresponding domain ontology concepts.

iii. **Matching phase**: allows mapping of entity from the documents with their corresponding concept in the domain ontology. The challenge is polysemy and synonym problem that should be tackled with the help of natural language processing. The matching process aims to map document elements to ontology concepts. The similarities between a document element and the concepts of the selected ontology will be computed to identify which concept will be attached to the initial element.

iv. **Annotation phase**: associates semantic metadata to the entities in the document through the process of annotation. Typically, since it is intended for use by humans and software agents, W3C recommended standard representation languages like RDF or OWL are used for semantic annotation representation.

v. **Evaluation phase:** aims to measure the quality of the document annotation by comparing with the gold standards. The main issue to be addressed when performing the semantic annotation of a document is the treatment of ambiguous, spurious and wrong annotations, especially when performing context-free semantic annotation. During this phase, completeness, expressiveness and semantic validity of the annotation will be evaluated to assure the quality of the annotation using precision, recall and, reliability methods. Therefore, this phase, defines how to measure quality against human-annotated gold standards in information extraction, such as correct (TP), missing (FN), spurious (FP), and partially correct (TN).

2.8.4 Semantic Annotation Approaches

Due to the vast and ever growing collection of available documents on the web, it is ideally impractical to manually annotate documents and great number of approaches on automatic semantic annotation have been proposed in the literature [9, 74, 81]. Thus, an automated annotation process provides major benefits including scalability needed to annotate existing documents, reduces the burden of annotating new documents, and allows for the use of multiple ontologies which can be beneficial to support the needs of different users.

Semantic annotation approaches can be classified into two primary categories [30] based on the their method of information extraction as "*pattern-based"* and "*machine learning-based"* as shown in Figure 2.7. In addition to the two categories, "multi-strategy" annotation approach is also used as a combination of the two techniques [83] in order to take advantage of their strengths, and compensate their weaknesses.

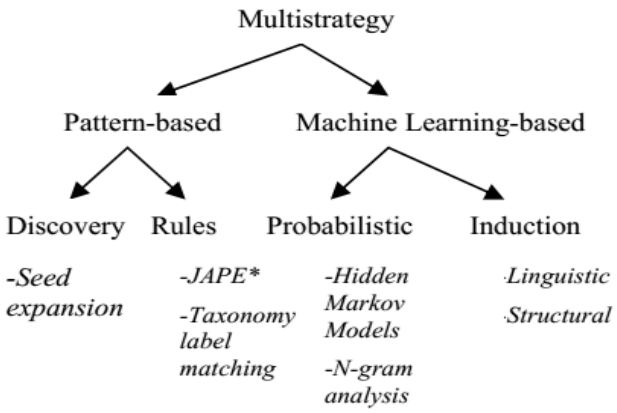

Figure 2.7: *Summary of Semantic Annotation Approaches*

a. Pattern-based Approach

Pattern-based semantic annotation can either apply pattern discovery algorithms, or have patterns manually defined by linguistic and domain experts [49, 69]. An initial set of entities are defined and a given corpus is scanned to find patterns in which the existing and new entities are discovered, along with new patterns. During semantic annotation, first initial set of entities are identified from the document using predefined patterns, then the initial entries are used to find rules based on the content of the entries. In this process, discovering new rules leads to discovering new entities and repeated with an expanded set of entities recursively until no more entities can be discovered or the user stops the annotation procedure.

b. Machine Learning-based Approach

As presented by Reeve [69], machine learning algorithms often perform more effectively than pattern-based methods for semantic annotation and other computational problems. Machine learning-based semantic annotation works utilize two methods: probability and induction. Probabilistic based semantic annotations use statistical models to predict the locations of entities within the text (e.g., Hidden Markov Models (HMMs)) [49]. A key attribute of HMMs is that they are adaptable in the case where information may be missing. There are also expanded work to build HMM models, in which there are two layers of an HMM rather than one. The upper level represents phrases, as before, but the lower level represents individual words within a phrase. For instance, domain-specific semantic search engines have been attempted to implement HMMs in order to perform semantic annotation. HMMs are more flexible than static rules due to their probabilistic nature [30], even though it has a disadvantage related to the amount of data required for training.

As stated by Reeve [30], semantic annotation research efforts for the web are not complete in their accuracy and requirement of the manual effort. The precision and recall of semantic annotation varies widely depending on the platform used, IE methods, and data source type. According to Reeve [30], HMM is a promising approach that will improve the performance of semantic annotation and reduce the required effort.

Table 2.4: *Examples of Different Annotation Strategies*

Approach	***Strategy***	***Example(s)***
Pattern based	Pattern discovery	Armadillo, Ont-O-Mat: PANKOW
	Rule-based	AeroDAML, GATE, MUSE
Machine Learning	Probabilistic	AutoBib, DATAMOLD
	Induction	Ont-O-Mat: Amilcare

However, since semantic annotation works normally have a knowledge base as a component of the system to link entities in a given resource with ontological concepts, it can be possible to develop a way to bootstrap the training process using information contained in the ontology and knowledge base. Therefore, the way to distinguish semantic annotation is based on extraction method used to find entities within a document as, whether or not machine learning is used, whether or not the platform allows manual rules, and whether or not the platform is

extensible. Examples of semantic annotation approaches are summarized as shown in Table 2.4.

2.8.5 Semantic Annotation Challenges

There are many challenges and obstacles in semantic annotation that lead to several research opportunities. Handschuh *et al.* [84] presented that manual semantic annotation has challenges of annotation consistency, proper referencing, redundancy avoidance, maintenance, ease of use and efficiency. In addition to these challenges, Hassanzadeh and Keyvanpour [85] have also classified challenges of semantic annotation systems into two inclusive classes: (1) general challenges, and (2) technical challenges. The general challenges category refers to those obstacles that exist regardless of technical and algorithmic considerations, such as multi-linguality and scalability problems.

On the other hand, the technical challenges contain the problems related to implementation and performance of semantic annotation system. Thus, an automated annotation method that provides scalability needed to annotate existing documents and reduce the burden of annotating new documents by allowing the use of multiple ontologies can be beneficial to the success of SW and also supports different web users need. This review of existing literature related to annotation systems indicated that semantic annotation research is an open area of research with multiple challenges. Nowadays, there are many semantic annotation research efforts which provide some of the requirements defined by Uren *et al.* [10]; however, fully integrated environment is still some way off and challenging.

2.9 Ontologies

Automated development of ontologies from text, ontology learning, has become increasingly important, because a large amount of texts for specific domains is already available in electronic format and using this textual data to learn background information is feasible in the case of the ontology learning. As reported by Badie [86], the act and role of learning can be identified as related to acquiring new or modifying existing knowledge. Thus, ontology learner components are concerned about the method of knowledge acquisition from texts in order to conceptualize a domain knowledge using the learning layer cake shown in Figure 2.8.

Ontology plays an important role in SW, especially in semantic annotation, aiming at finding semantic correspondences between document entities and ontological concepts. Knowledge-

based systems also grew larger and the commercial interest in these technologies increased from time to time. However, the process of creating an ontology is complex, involving many hours of manual work and requires existing knowledge of domain. Nowadays, people became aware of the knowledge acquisition bottleneck and the necessity to (partly) automatize the creation and maintenance of knowledge bases, as a result, any tool that might help the process of ontology construction through automation is at ease for different knowledge based systems.

Nowadays, the web has become the largest repository of knowledge, it is open and allows anyone with the Internet access to add or use the knowledge. Previous studies indicated that numerous approaches have been proposed to automatically construct ontological concept relationships from collaborative data and the first type of collaborative data comes from Wikipedia. DBpedia [87] and YAGO [88] are the most famous systems for handling such data and both systems use structured information on Wikipedia which include links, categories, and info-boxes.

2.9.1 Ontology Graph

Ontology graph (OG) is an approach used in [89] to model the ontology of the domain from text. The OG consists of different levels of conceptual units, in which they are associated together with different kinds of relations. It is basically a lexicon system (terms) that linked up among each other to represent a group (a cluster), to formulate concepts and represent meanings. The conceptual structure of an OG consists of many terms with some relationships between them, so that different conceptual units are formed like a network model [89]. An individual term node in the OG is the most basic conceptual unit represented by a single term, a meaningful term (a sequence of characters), which contribute to defined concepts. To define a lexical word as a term node in OG, it is needed to select a word that is "*meaningful*" [89]. In natural language system, the four basic grammatical categories of words are: noun, adjective, adverb, and verb. Other terms, including adverb are filtered and excluded from the OG, due to their no (or less contribution) in ontology graph creation [89].

Multiple term nodes in OG grouped in a cluster and forms a concept node. Thus, concept node is formulated by any cluster of nodes with relations, representing a certain concept by grouping semantically similar terms together. The OG consists of four types of conceptual units (CU), any objects (nodes) in OG that give semantic expression, according to their level

of complexity exhibited in domain knowledge. CUs are linked up and associated with conceptual relation within each other, to comprise the entire conceptual structure of OG, and to model domain knowledge. The four CU definitions, their natures and the levels of knowledge according to their complexity are described as follows.

- **Term**: The smallest conceptual unit that is extracted in the form of a meaningful word (a sequence of characters), consist of "*meaning*" in human perspective.
- **Concept**: A number of term grouped together with conceptual relations between each other and it is the basic conceptual unit of the concept graph.
- **Concept Cluster**: A number of concepts related to each other and form a concept cluster. It groups similar meaning of concepts in a tight cluster representing a hierarchy of knowledge.
- **Ontology Graph**: The largest, entire conceptual unit grouped by concept clusters and represents a comprehensive knowledge of a certain domain.

In semantic web applications, ontologies describe the formal semantics that make existing web information machine-understandable and also are used to support interoperability between different applications in a way that solves the problem of semantic heterogeneity. As presented by Navigli and Velardi [90], ontologies play a central role in a knowledge-based semantic technologies such as semantic search, intelligent information integration, semantic annotation, and semantic based NLP. Thus, ontologies are serving as a backbone for different semantic based applications, and remarkably rare in less resourced languages, which can be a challenge for the wider fields such as semantic annotation of web documents. Thus, in the context of semantic web, an ontology is a vital element for semantic annotation of web documents.

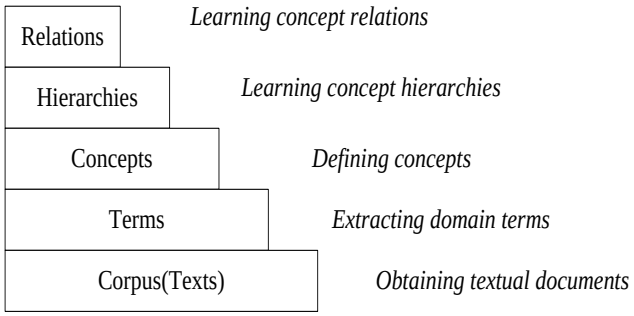

Figure 2.8: *Knowledge Components of Ontology Learning from Text*

The study of ontology is very broad and covers many tasks including ontology generation, enrichment, merging, and mapping. As depicted in Figure 2.8, ontology learning from textual corpus is mainly organized into five sub-components [89] from obtaining unstructured textual corpus up to relation extraction.

2.9.2 Ontology Modeling

Lemon (Lexicon Model for Ontologies) is a model for associating linguistic information with ontologies, in particular for SW ontologies. Lemon is a flexible model that extends SW in new ways and leads to much needed resources for different NLP applications depend on ontologies and semantic web [91]. Borin *et al.* [92] stated that the use of lemon has already proved to be a key component in systems for information extraction that will help us for information extraction of semantic annotation. Ontology languages like XML, DAML+OIL, RDF(S), and OWL allow users to write explicit, formal conceptualizations of domain models in ontology construction and they summarized as follows:

i. **XML** provides syntax for structured documents, but imposes no semantic constraints on the meaning of documents.

ii. **DAML+OIL** is ontology language created after merging ontology interchange language (OIL) with darpa agent markup language (DAML). This language divides the universe into two disjoint parts named as a datatype and object domain.

iii. **RDF-Schema (RDF(S))** is a language for defining vocabulary for describing properties and classes of RDF resources. RDF(S) is used to define graphs of trio RDF, with the semantics of generalization of such properties and classes.

iv. **Web Ontology Language (OWL):** Building upon RDF and RDFS, OWL provides more machine-interpretable semantics by defining additional vocabulary along with a formal semantics. OWL builds on description logics (DLs) which is a restriction of first order logic. OWL provides three increasingly expressive sublanguages: OWL Lite, OWL DLs and OWL Full. Each of these sublanguages are an extension of its simpler predecessor. Compared to the other two sublanguages, OWL DL is often chosen as the ontology modeling language because of its capacity of fair semantics expressiveness and inference. As noted by Abdelrahman and Kayed [43], using OWL

Full compared OWL DL, reasoning support is less predictable, since the full implementation of OWL Full does not currently exist.

Most of the ontology languages are supported by different tools (like: Apollo, Protégé, and Swoop) which are open-source and suitable for ontology development by providing visual based functionalities for ontology manipulation, inspection, browsing, coding, and development [93]. These tools are widespread in ontology design and development sector and accepted by large semantic web communities.

i. **Apollo**[13] is an open source and user friendly knowledge modeling tool that allows user to model ontology with basic primitives such as classes, instances, functions, and relations. Apollo does not support graph view and multi-user capabilities, but it has consistency checking and import/export plugin, which stores ontology in file format.

ii. **Protégé**[14] is an open-source ontology editor that allows definition of classes, class hierarchy's variables, variable-value restrictions, and the relationships among classes and properties of these relationships [94]. It supports import of textual files, database tables and RDF files as an input for further ontology manipulation.

iii. **Swoop** is an open-source, Web-based OWL ontology editor and browser developed by the University of Maryland. It contains OWL validation, reasoning support, and provides a Multiple Ontology environment, by which entities and relationship across various ontologies can be compared, edited and merged seamlessly.

2.9.3 Ontology Learning

Ontology learning (ontology extraction, ontology generation, or ontology acquisition) is a task of extracting relevant concepts and relations from a given corpus or other kinds of datasets to create an ontology [93] as a model of the world knowledge using computational techniques. Research on automated development of ontologies from text has become increasingly important, because manual construction of ontologies is labor intensive and at the same time, a large amount of texts for specific domains are already available in electronic formats.

[13] It is written in Java programming language and freely available for download at: https://apollo.open.ac.uk
[14] It has full supports for latest OWL 2 ontology language and RDF specifications: https://protege.stanford.edu

In ontology learning process, there are two fundamental aspects. The first one is the availability of existing knowledge, in other words, whether the learning process is performed from the scratch or some existing knowledge. In this scenario, the initial version is then extended automatically through learning procedures. The other aspect is the type of input used by the learning process. OL can use unstructured (texts in natural language), semi-structured (dictionaries) or structured (database schemas) data as an input for the sake of learning. As discussed by Ribeiro [23], ontology learning processes mostly rely on unstructured texts, since most of the knowledge sources in the web are in unstructured format. In semantic web context, OL is primarily concerned with knowledge acquisition from the web contents and is thus moving away from small and homogeneous data collections to tackle massive and heterogeneous data of the WWW [20].

So far, several efforts have been devoted to the study of ontology learning as the process to learn and create domain knowledge in a particular area of interest. As an approach, a bottom-up ontology learning approach that identifies and generates conceptual units from the lowest level, term (T), to the highest level, the ontology graph (OG) are often used in domain knowledge modeling from unstructured text. The feasibility and effectiveness of this learning approach has been tested for different languages and domains [89, 95, 96, 97]. As presented in [10], although the researchers have studied many different ways to generate ontologies automatically, semantic content organization and retrieval engine (SCORE) is the only system that has been tried as automatic ontology learning for semantic annotation.

Ontology Learning Phases

As it can be seen from Figure 2.8, ontology learning tasks called "*Ontology Learning Layer Cake*" organizes tasks in increasing complexity of meaning that more complex tasks rely on the output of less complex tasks. At the lower end of the layer cake, terminology based tasks like "acquisition of the relevant terminology" and "acquisition of synonym terms / linguistic variants" that can be followed by the identification and organization of concepts and relations.

i. Term Extraction

The first step of the ontology learning process is to extract terms that have great importance to describe a domain. It is highly relevant for the process of ontology learning to know which words in a set of documents are most representative for that set. Two popular techniques for co-occurrence analysis are TFIDF and Pearson's χ^2.

ii. Concept Definition

The second step in OL is to determine which of the extracted terms are used to define domain concepts. According to Buitelaar *et al.* [98], term can represent a concept if we can define: its intention (giving the definition, formal or otherwise, that encompasses all objects the concept describes), its extension (all the objects or instances of a given concept) and to report its lexical realizations (a set of synonyms in different languages). In ontology learning, a considerable amount of time is devoted to the initial phase that involves the selection of ontological concepts from the data source.

As presented by Lim *et al.* [89], aiming at minimizing this issue, concept extraction and definition techniques use technologies derived from natural language processing and data mining field of computation. However, the lack of such resource for all languages still hinders the automation of creating ontologies from less resourced language's unstructured text. In concept extraction, if the corpus of relevant documents is only text, the notion of the concept would be equivalent to the word. As reported Jacobs and Monachesi [99], nouns are linguistic elements most often used as a concept.

In this sense, concept can be single or multi word nouns in textual corpora that could be relevant for ontology construction. Thus, in ontology learning process, concepts that are of great importance to describe a domain as a basic semantic unit have to be extracted from the corpus. Moreover, definition based synonymity extraction is also required under concept definition phase by grouping terms referring to the same real world entities. According to theory of definition by Aristotle, the formal structure of definition should look like an equation with the definiendum (i.e., what is to be defined) on the left hand side and the definiens (i.e., which is doing the defining) on the right hand side. The definiens also consisting two parts: the nearest superior concept and the distinguishing characteristics.

iii. Taxonomy Induction and Relation Extraction

In ontology learning, automatic taxonomy induction is an important task in the fields of NLP, knowledge management, and semantic web. Existing works on automatic taxonomy induction have been conducted under variety of names, such as ontology learning, semantic class learning, semantic relation classification, and relation extraction. The

approaches used for taxonomic extraction basically fall into two categories named as pattern based and clustering based. In pattern based approaches, set of lexico-syntactic patterns have to be defined, which are applied to the texts in order to obtain instances of taxonomic relations. In clustering based approaches, hierarchical clustering algorithms are used for finding the taxonomic relations between concepts. Various techniques have been presented in the literature for the learning of semantic relations among concepts ranging from association based relation extraction up to knowledge based relation extraction.

iv. Axiom

The extraction of axioms is the final level of the learning process and it is argued to be the most difficult one [6, 7]. To date, few research efforts have been attempted for the discovery of axioms and rules from text.

Ontology Evaluation

Ontology evaluation is the task of measuring quality of ontology, since effective semantic annotation requires well qualified ontologies in order to achieve quality semantic annotation. Gangemi *et al.* [100] stated that ontologies, like all engineering artifacts, need a thorough evaluation, but the evaluation of ontologies poses a number of unique challenges: due to the declarative nature of ontologies in which developers cannot just compile and run the learning like most other software artifacts. Gangemi *et al.* [100] have proposed schema validation, pattern discovery using SPARQL, normalization, metric stability, representational misfit, and unit testing methods for ontology evaluation.

Chapter 3

Related Work

3.1 Introduction

Several research efforts have been conducted and proposed for automatic semantic annotation; however, they are not implementation clones of one another, since each work is designed to address different annotation needs and objectives. Thus, this chapter presents related works conducted in the area of semantic annotation to address document semantic tagging related problems. Section 5.1 presents some frameworks for semantic annotation. Section 5.2, 5.3 and 5.4 present semantic annotation related works used pattern, ML, and multi strategic approaches. Section 5.5 presents the identified requirements required for the success of semantic annotation, but not addressed in current related works. Finally, section 5.6 presents the summary of related works with existing gaps.

3.2 Semantic Annotation Frameworks

Current research on semantic annotation is trying to match heterogeneous requirements that lead to divergent methodologies, models and processes for semantic annotation management and exchange. Accordingly, several comprehensive frameworks for different semantic annotation requirements have been proposed so far, and under this sub-section BNOSA, Annotea, CREAM, and Arabic semantic annotation framework are presented.

3.2.1 BNOSA

Bayesian network and ontology based semantic annotation (BNOSA) [101] is a semantic annotation framework that is capable of extracting relevant information from unstructured and incoherent data sources. As it can be seen from Figure 3.1, BNOSA utilizes ontology and Bayesian networks to perform information extraction and semantic annotation tasks. The framework is extensible as it is capable of dynamically extracting data from any domain with pre-defined ontology and corresponding Bayesian networks. The sets of corpora used in their experiments belong to "*selling–purchasing*" websites, where product information is entered by ordinary web users in a structure-free format. The result of the conducted experiment shows that BNOSA performs reasonably well to find the location of the data of interest using context keywords provided as part of domain ontology. In case of more than one value being extracted

for an attribute or if the value is missing, Bayesian networks identify the most appropriate value for that attribute. In line with this, ontology based semantic annotation of Urdu language web documents [76] is also an extension of BNOSA and presented as semantic annotation framework for unstructured and ungrammatical content in Urdu language in addition to English language content. The annotation approach is based on domain ontology that makes it easily extendable by adding other domain ontologies.

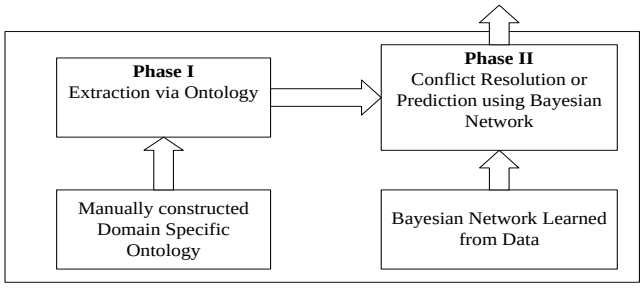

Figure 3.1: *Components of BNOSA Framework*

3.2.2 Annotea

Annotea [102] is a W3C project which specifies infrastructure for annotation of web documents. The main format for Annotea is RDF and the kinds of documents that can be annotated are limited to HTML or XML-based documents. It provides XPointer as a ready-made structure for locating annotations within a document. XPointer is a W3C recommendation for identifying fragments of URI resources. The authors presented that as long as the component of a document to which an XPointer refers is retained, the location of the associated annotation is robust to changes in the detail of the document. However, if large scale revisions are made, annotations can easily come adrift from their anchor points.

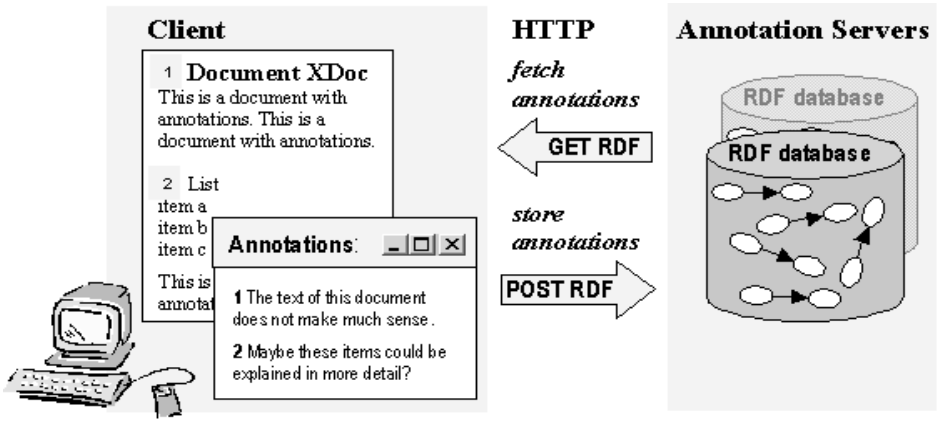

Figure 3.2: *Architecture of Annotea*

The Annotea approach concentrates on a semi-formal style of annotation, in which annotations are free text statements about documents. These statements must have metadata (author, creation time, etc.) and may be typed according to user-defined RDF schema of arbitrary complexity. In this respect, Annotea is not quite as formal as would be ideal for the creation of intelligent documents and its workflow is depicted in Figure 3.2. In Annotea, annotations being stored as RDF held either on local machines or on public RDF servers. Examples of tools based on the Annotea framework are Amaya and Annozilla. Amaya is a good example of a single point of access environment which has facilities for manual annotation of web pages, but it does not contain any features to support automatic annotation.

3.2.3 CREAM

Creating relational annotation-based metadata (CREAM) [84] is a comprehensive framework that looks at context in which annotations could be made for creating metadata. This framework allows relational metadata, defined as "annotations which contain relationship instances". Handschuh *et al.* [84] presented that relational metadata is essential for constructing knowledge base which can be used to provide semantic services. Figure 3.3 depicts the overall components required by CREAM annotation system, including annotation interface, with automatic support for annotators, document management system and annotation inference server.

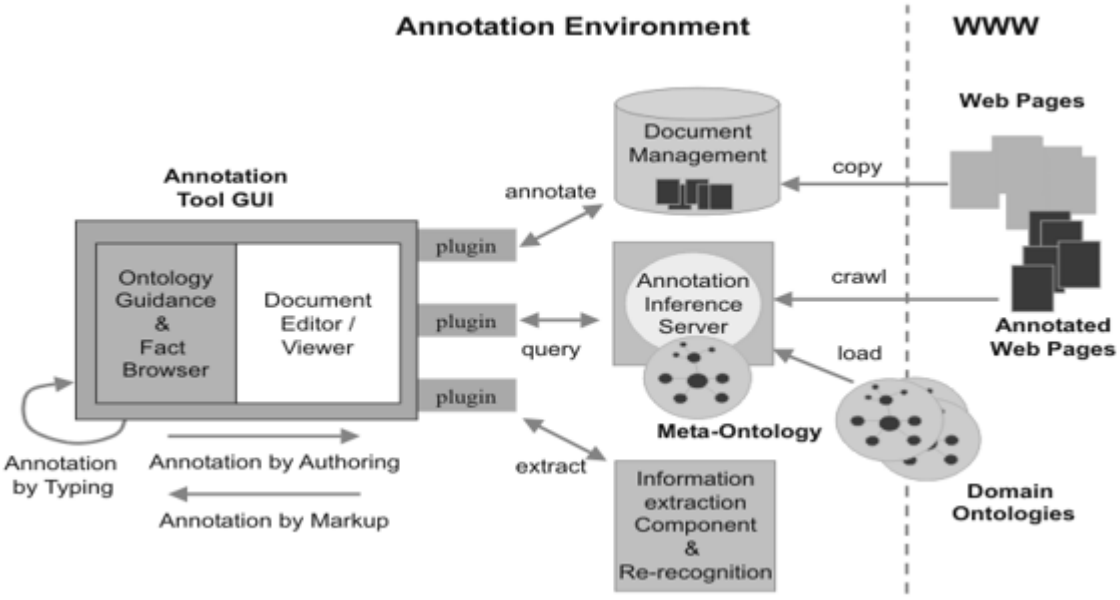

Figure 3.3: *Architecture of CREAM*

CREAM annotations made in RDF, and XPointer is used to locate annotations in text. It is restricted to web-native formats such as XML and HTML. Unlike Annotea, Handschuh *et al.* [84] considered the possibility of annotating deep web. It is supported by a storage model that allows users to choose whether they want to store annotations separately on a server or embedded in a web page. Some examples of tools based on the CREAM framework are OntoMat-Annotizer and S-CREAM.

3.2.4 Arabic Semantic Annotation Framework

El-ghobashy *et al.* [19] proposed Arabic semantic annotation framework that requires four major components, which are, a text preprocessing module, information extraction module, semantic annotation module, and annotation management module. The purpose of this framework is to enhance browser (e.g., Firefox or Google Chrome) functionalities to handle annotations of a web page. Their work does not consider document annotation consistency issue. Figure 3.4 shows the framework proposed for Arabic document semantic annotation.

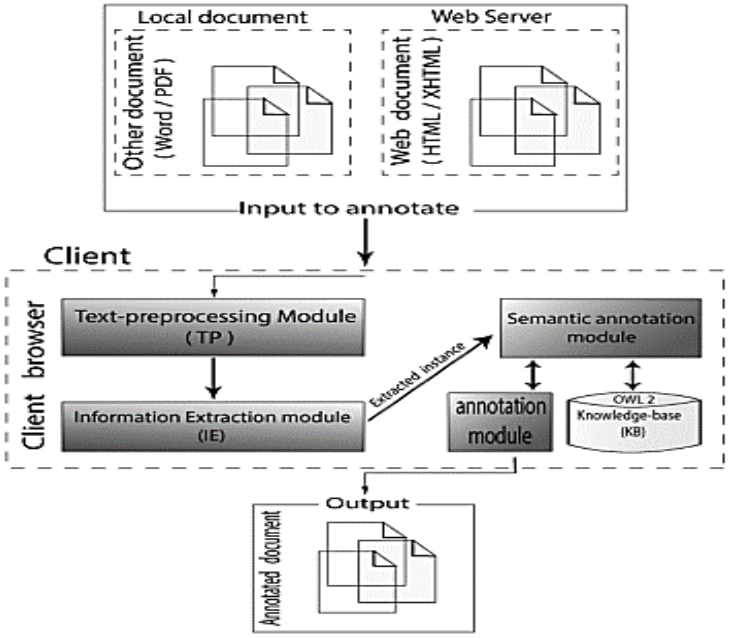

Figure 3.4: *Framework for Arabic Document Semantic Annotation*

3.3 Pattern based Semantic Annotations

Pattern-based semantic annotation either include pattern discovery mechanism or manually defined patterns. KIM, SemTag, ONTEA, and other related pattern based semantic annotations are discussed under this section.

3.3.1 KIM

Knowledge and information management system (KIM)[15] [17, 103] is an extensible knowledge management for semantics-based support that offers information extraction based facilities for metadata creation, storage and conceptual search. This annotation project is a part of SWAN[16] project that consists of a formal ontology, knowledge base, information extraction, indexing, retrieval, front-end, and server that provide full access to the functionality of the KIM. Figure 3.5 shows the semantic information extraction flow of KIM, it focuses on NEs, such as people and location that are referred by name, mentioned within a document as an important constituent of document semantics. A semantic annotation is based on this hypothesis and the annotation is done by assigning links between entities in the text and their semantic descriptions.

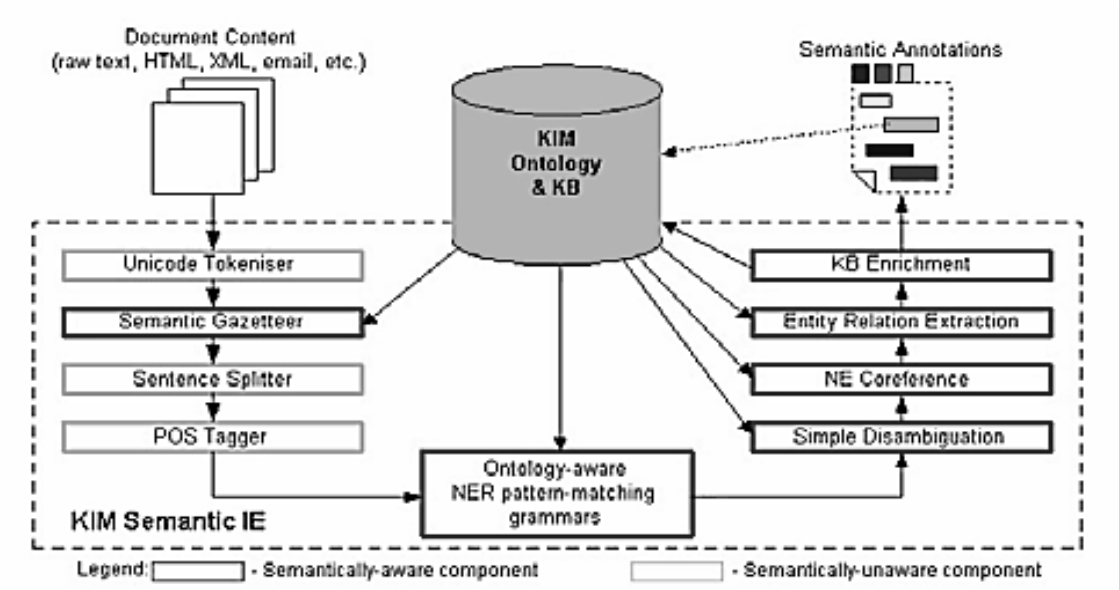

Figure 3.5: *KIM Semantic IE Flow Diagram*

In KIM information extraction is performed based on upper level ontology and a massive knowledge base. The ontology used for KIM based semantic annotation is called KIM Ontology (KIMO), a light-weight upper level ontology that contains definitions of entity

[15] http://www.ontotext.com/KIM

[16] SWAN, Semantic Web Annotator is an experiment in scaling up automated metadata extraction for industrial strength Semantic Web applications development.

classes, attributes, and relations, and also a branch of lexical resource types. The knowledge base of KIM is used for keeping semantic descriptions of entities and relations between them. For each entity reference in the document, KIM provides two links: one is a link to the most relevant class in the ontology and another is a link to the specific instance in the knowledge base. The authors chose RDF (S) as their ontology representation language. During the annotation process, KIM employs NLP information extraction techniques, which is based on GATE[17] framework that enables users to develop and deploy language engineering components and resources in a robust fashion. As stated by the authors, KIM used different GATE's document management functionalities and generic NLP components like Tokenizer, POS tagger, and sentence splitter in order to extract, index, and annotate data instances. KIM was tested and resulted with good accuracy indicated as average recall of 84%, precision of 86%. KIM does not used advanced IE techniques for identification of relations, analysis of events and also has limitations regarding crawling websites and annotating web pages. KIM annotation is restrictive, and it would be a significant research challenge to extend the current KIM methodology to domain specific ontologies.

3.3.2 SemTag

SemTag [18] is an automated semantic tagging component of a comprehensive platform, called Seeker, for performing large-scale annotation of web pages which is a part of WebFountain[18] research. Seeker is a platform which supports large-scale text analytics and performs the annotation process in three passes, namely spotting pass, learning pass, and tagging pass. Spotting pass includes retrieving and tokenizing documents from the seeker store and processed to find label matches from TAP taxonomy covers a range of lexical and taxonomic information about popular items such as music, movies, authors, sports, and health. The learning pass includes scanning representative sample of the corpus in order to find the corpus-wide distribution of terms at every node of the taxonomy. Finally, in the tagging pass, windows based scanning is followed by matches' disambiguation, when a label and an actual TAP object are found matching. Finally, URL, reference and other metadata are entered into the database as final results. SemTag uses a taxonomy based disambiguation algorithm (TBD)

[17] General architecture of text engineering, http://gate.ac.uk/

[18] Web fountain is a web scale mining and discovery platform that combines breakthrough text analytics technology and very large, heterogeneous data sources with custom solutions.

as shown in Figure 3.6 for resolving ambiguities using vector space model as a technique. VSM is used either to assign correct ontological class or determine a concept that does not correspond to class in TAP in order to overcome disambiguation problem. SemTag system is implemented on a high-performance hardware with parallel architecture, where each node annotates about 200 documents per second. SemTag has a pool of approximately 72,000 labels used to find entity instances within a text and incorporates an algorithm as part of its tagging process to determine the probability of a particular label belonging to a particular class in TAB ontology. SemTag does not attempt to discover and classify new instances, which are not already in the TAP ontology.

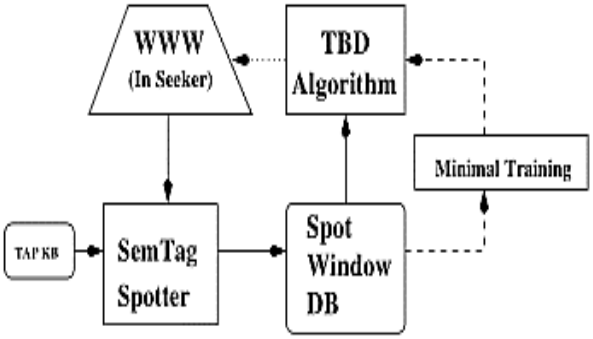

Figure 3.6: *SemTag Architecture*

SemTag was evaluated on a set of 264 million web pages, the tool was able to generate and disambiguate 550 million semantic tags, and approximately 79% of them were judged to be on-topic. During this experiment 750 human judgments were used as a training set for the algorithm and other 378 human judgments were applied to estimate the performance. The system was executed on 128 dual processor 1GHz machines and the total time taken to process the web was 32 hours.

3.3.3 ONTEA

ONTEA, ontology based text annotation [104], is a semi-automatic ontology based text annotation that was created as a part of NAZOU[19] project. Inputs to ONTEA are textual resources (like: HTML, email, or plain text). In ONTEA, text or a text document is analyzed using regular expression patterns and equivalent semantic elements are detected using domain

[19] http://nazou.fiit.stuba.sk/

ontology. Ontology individuals corresponding to the annotated text are generated as output. The detected ontology individuals are then used to fill the properties of extracted individuals from textual document using predefined patterns for extraction. ONTEA uses RDF/OWL format of ontologies and its implementation was carried out in java using Jena semantic web library and sesame library.

3.3.4 OntoAnnotate

OntoAnnotate [105] can be regarded as a workbench for semantic annotation of documents using domain-specific ontologies. The purpose of OntoAnnotate is to enrich HTML pages with semantics. This enrichment enables software agents capable to automatically process and understand content of web pages and reasoning about it. In identifying entities in web pages, it uses a combination of the following techniques: wrapper generation, pattern matching and ontology based information extraction based on a shallow text processing engine.

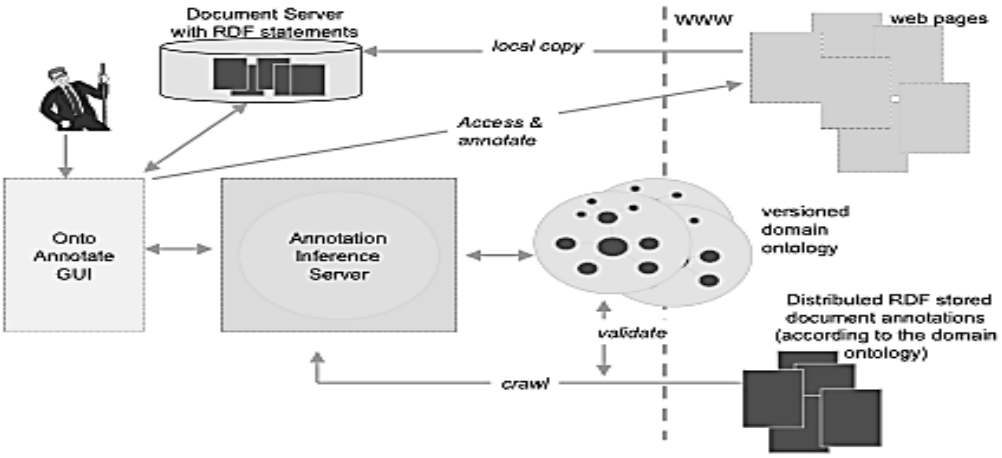

Figure 3.7*: OntoAnnotate Semantic Annotation Environment*

This tool makes relationship between particular ontologies and their parts (i.e., concepts and properties) explicit. The core of OntoAnnotate is used for viewing web pages and actually providing annotations using shallow text processing for German. Figure 3.7 shows the overall annotation environment of OntoAnnotate.

3.3.5 PANKOW

PANKOW, pattern based annotation through knowledge on the web, is an annotation method which employs an unsupervised pattern based approach. It is called unsupervised because it

does not rely on any training data annotated by hand and pattern-based because it makes use of linguistically motivated regular expressions in order to identify instance-concept relationships from a given text. It works by categorizing instances (candidate noun phrases) with regard to a given ontology. It is implemented in Ont-O-Mat which is an annotation tool used for the semantic web. The annotation process in PANKOW consists of the following steps shown in Figure 3.8, first a web page is given as an input and POS tagger scans the inputted web page for candidate phrases extraction that can be categorized as instances of ontology classes. Next to this, the system derives hypothesis phrases by using candidate proper nouns and all candidate ontology concepts. Google is queried for the derived hypothesis phrases through its web service API to get number of hits for each hypothesis phrase. Candidate proper nouns are then categorized into their highest ranked concepts. Hence, the system annotates a piece of text as describing an instance of that concept, and ontologically annotated web page is generated as an end result.

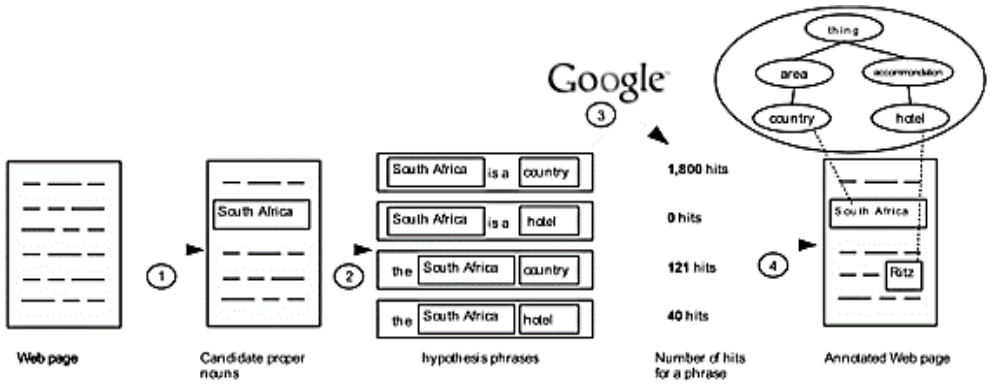

Figure 3.8: *The Process of PANKOW*

PANKOW has an integration into the CREAM framework and extends the CREAM implementation with Ont-O-Mat that supports two modes of interaction named as fully automatic and interactive semi-automatic. PANKOW does not rely on a seed corpus, since it makes use of linguistically motivated regular expressions to discover relations in the document. It uses the knowledge existing in the web to propose annotations based on counting Google hits of instantiated linguistic patterns.

3.3.6 Automatic Ontology based Annotation for Arabic Web Content

Automatic ontology-based annotation of food, nutrition and health Arabic web content [106] is the annotation tool that starts with web resources acquisition and ends up with the RDF file.

The RDF file stored in semantic repositories which could be used by semantic web technologies. The annotation process consists of seven main tasks: web source acquisition, tokenization, normalization, NER, fact extraction, fact cleaning and validation, ontology mapping and knowledge based enrichment as shown in Figure 3.9. The tool was developed using Java with an embedded version of the GATE NLP toolkit.

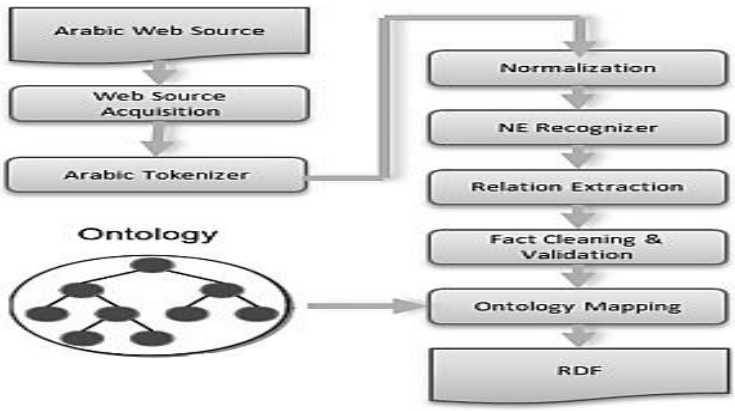

Figure 3.9: *Annotation Process for Arabic Web Content*

An advantage of all pattern based semantic annotations are the advantage of having pattern or rule that do not requires text mining and learning. However, it is expensive to scale and maintain the pre-defined pattern. Moreover, each time a data source changes, the pre-defined patterns may also need to be changed. Thus, language specific resources and patterns are expensive to adapt into new languages and domains.

3.4 Machine Learning based Semantic Annotations

Machine learning-based semantic annotation employ probability and induction techniques in their annotation process. They use probability such as statistical models to predict location of entities in a textual document, and induction to link the extracted entities using probabilistic and other ML techniques into ontological concepts.

3.4.1 MnM

MnM[20] [16, 31, 107] is an ontology-based annotation tool, which integrates web browser, ontology editor and open APIs to provide both automatic and semi-automatic supports for annotation of texts in a web page. It provides an environment to manually annotate a training

[20] http://projects.kmi.open.ac.uk/akt/MnM/

corpus, and then feed the corpus into a wrapper induction system based on Lazy-NLP algorithm (i.e., an algorithm based on linguistic information, and rules are generated using sets of conjunctive conditions on adjacent words). Once the system is trained and rules are induced from a training corpus, the system can begin to extract information from source documents and returns annotated documents. MnM is a combination of ontology servers, information extraction tools and augmented web browsers and it works by integrating the web browser with an ontology editor. It offers an open API for connecting to ontology servers and combining the tools of information extraction. The working of MnM includes mainly five activities, namely: *browse, markup, learn, test* and *extraction*. A library of knowledge models from web is browsed and specific knowledge set is selected by the user. In MnM, handcrafted KMi ontology is used for annotating the documents with a set of tags. Learning phase of MnM makes use of Amilcare and Annie for enabling information extraction through the learning of extraction rules in the form of tagging rules and correction rules. Finally the induced rules are used for extracting information from texts. MnM does not separate the annotations from the annotated document and it was designed to markup training data for IE tools rather than as an annotation tool.

3.4.2 Semantically Annotating Corpus using Machine Learning

Semantically annotating corpus using machine learning [108] is the proposed methodology exploits domain ontology and NER system automatically trained in the specific domain using Hidden Markov Models (HMMs). The ontology knowledge is used for initial annotation of the corpus with ontology instances. This corpus is then used to train a NER system using a machine learning method, HMMs. The resulting NER system will be then able to identify new named entities that are not included in the ontology. The contribution of the HMM-based NER system is important, as it can identify named entities not included in the ontology. The ontology is also important, as it corrects some entities erroneously identified by the NER system. The authors reported that their experimental results highlight the effectiveness of collaborating different knowledge sources.

3.4.3 SARM

SARM [109] is a semantic annotation system which has automatic instance extraction module based on two machine learning techniques, Bayesian and support vector machine (SVM)

classifier and its architecture is shown in Figure 3.10. SARM consists of automatic instance extraction module (web document crawler, web RawDB, HTML parser, morphology analyzer, Bayesian classifier and SVM classifier), semantic annotator (with API for remote access, embedding, and integration), domain ontology, annotated results (RDF or OWL statements for semantically annotated web contents) and front-ends (user interface with web browser control and knowledge explore for ontology navigation). The performance of SARM was evaluated by accuracy micro average after conducting 5-fold cross validation on 1,260 extracted restaurant domain instances. The training and testing sets used for the 5-fold cross validation consists of 1,008 and 252 restaurant instances respectively.

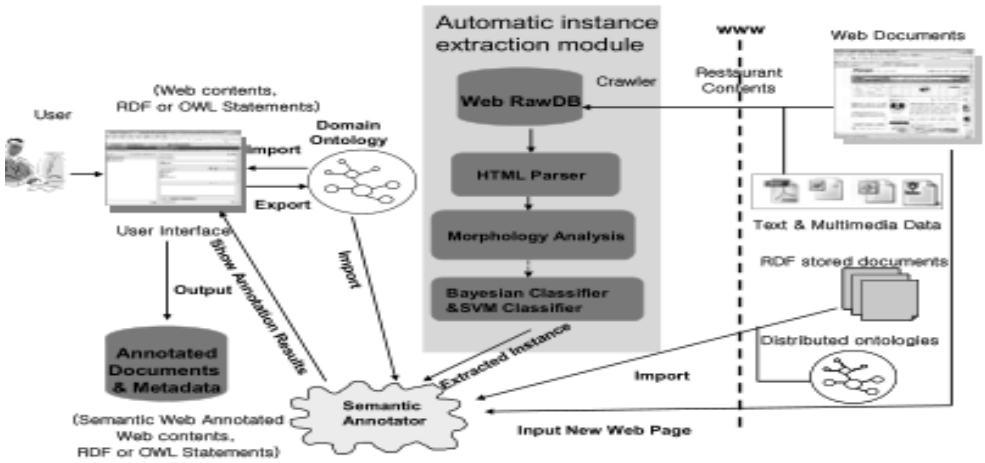

Figure 3.10*: Architecture of the Korean Document Semantic Annotation*

Semantic annotation will take advantage of machine-learning based approaches, because machine learning help to relieve the manual effort required in building rules, which is the major drawback of pattern-based approaches. However, studies on machine learning based semantic annotation mostly using classical machine learning techniques that works well because of human-designed representations and features that also have major drawbacks in terms of requiring expensive training data. The classical ML also prone to over fit to the training corpus that makes results of evaluations very hard to extrapolate to other texts, specifically for morphologically complex languages.

3.5 Multi Strategy based Semantic Annotation

A multi-strategy approach uses a combination of both machine-learning and pattern-based approaches. To the best of our knowledge, no semantic annotation to date is using a complete

multi-strategic approach incorporating both pattern and machine-learning approaches. MUSE [110] system comes as the closest by using text features and then conditionally executing rules based on the text features. MUSE was designed to perform named entity recognition and co-referencing and its implementation using the GATE framework.

3.6 Requirements

In addition to requirements surveyed and defined by Uren *et al.* [10], there are also other requirements need to be considered in development of semantic annotation technologies. We consider these requirements as drawbacks of existing related works.

i. **Structure Analysis:** document structure analysis is fundamental for document processing and understanding. Thus, the way we are representing a document is an issue that has to be considered, since most of existing methods representing document as a "*bag of words*" and misses the structural issue.

ii. **Relation Extraction:** Many applications in information extraction, natural language understanding, and information retrieval require an understanding of the semantic relations between entities, on the contrary less attention is given for non-taxonomic relation that has great importance for the research work related to semantic annotation.

iii. **Ontology Learning Integration:** Semantic annotation requires ontologies and knowledge bases to perform instance to concept mapping. However, ontology learning integration, is poorly supported, or not supported at all, by the current generation of semantic annotation approaches. There is a gap in managing difference between document content and ontology classes and properties due to a premodeled nature of an ontology.

iv. **Domain Adaptation and Scalability:** Semantic annotation methods can learn good annotation models for a specific domain. However, the annotation models adaptation into other domains with no (minimal) modification is challenging due to the domain specific nature of current systems. Thus, considering domain issues is a key factor to the development of a real-world semantic annotation system. Semantic annotation tasks must be coordinated in a scalable manner, since the solutions should consider huge growth of the existing web. For instance, in pattern based annotation, the limiting factor on the scalability of such systems is related to the expensive rule generation and maintenance process. Each time a data source changes, the pre-defined rules may also

need to be changed and this hinders scalability of semantic annotation and other semantic web technologies.

v. **Access Controlling:** Document has to be tagged with privacy information depending on privacy requirements of the owner and content of the document. Thus, access controlling mechanism is in need to prevent the semantic annotation from unauthorized software agents and users. However, there is no consideration for privacy issues in the related work we have reviewed under this chapter.

vi. **Granularity (Detail) Level:** There are different levels of granularity (like: paragraph, sentence, concept, term or word) to understand document. However, existing systems for semantic annotation (e.g., KIM and MnM) are only capable of annotating document at word or term level. None of them provide annotation on granularity level above word or term at paragraph or sentence level. Thus, sentence and paragraph level annotation is required to understand and annotate web documents with better and richer information.

3.7 Summary

To sum up, existing semantic annotation research efforts varied in their architecture, information extraction method, initial ontology, manual work required to perform annotation, and other supporting features (such as storage management). These differences have made a challenge of comparing semantic annotation systems only using their performance results and Annex A provides comparison of related works with their drawbacks. The reviewed related works have also drawbacks in supporting all requirements defined by Uren *et al.* [10] and others discussed under Section 3.6. Moreover, our review also shows that although there are many research works available, they focus on classical methods of information extraction; however, generality and advanced information extraction methods consideration are also another important dimension that have to be considered in machine learning based semantic annotation. Due to less attention for generality and advanced machine learning techniques, success of semantic annotation for less resourced languages lagging behind and hinders success of semantic web technologies. By considering the stated requirements and challenge of adapting existing works, the proposed new semantic annotation framework for web documents and its detail are discussed in the next chapter.

Chapter 4

Generic Semantic Annotation Framework

4.1 Introduction

To achieve semantic web goal, software agents must be equipped with sophisticated semantic analysis and understanding of web documents to publish semantic information in a form accessible to other software agents. Thus, this chapter presents the proposed generic framework for automatic document semantic annotation by describing different components along with relevant techniques and proposed algorithms. Figure 4.1 depicts the overall design of the framework and details of each component is discussed under section 4.2.

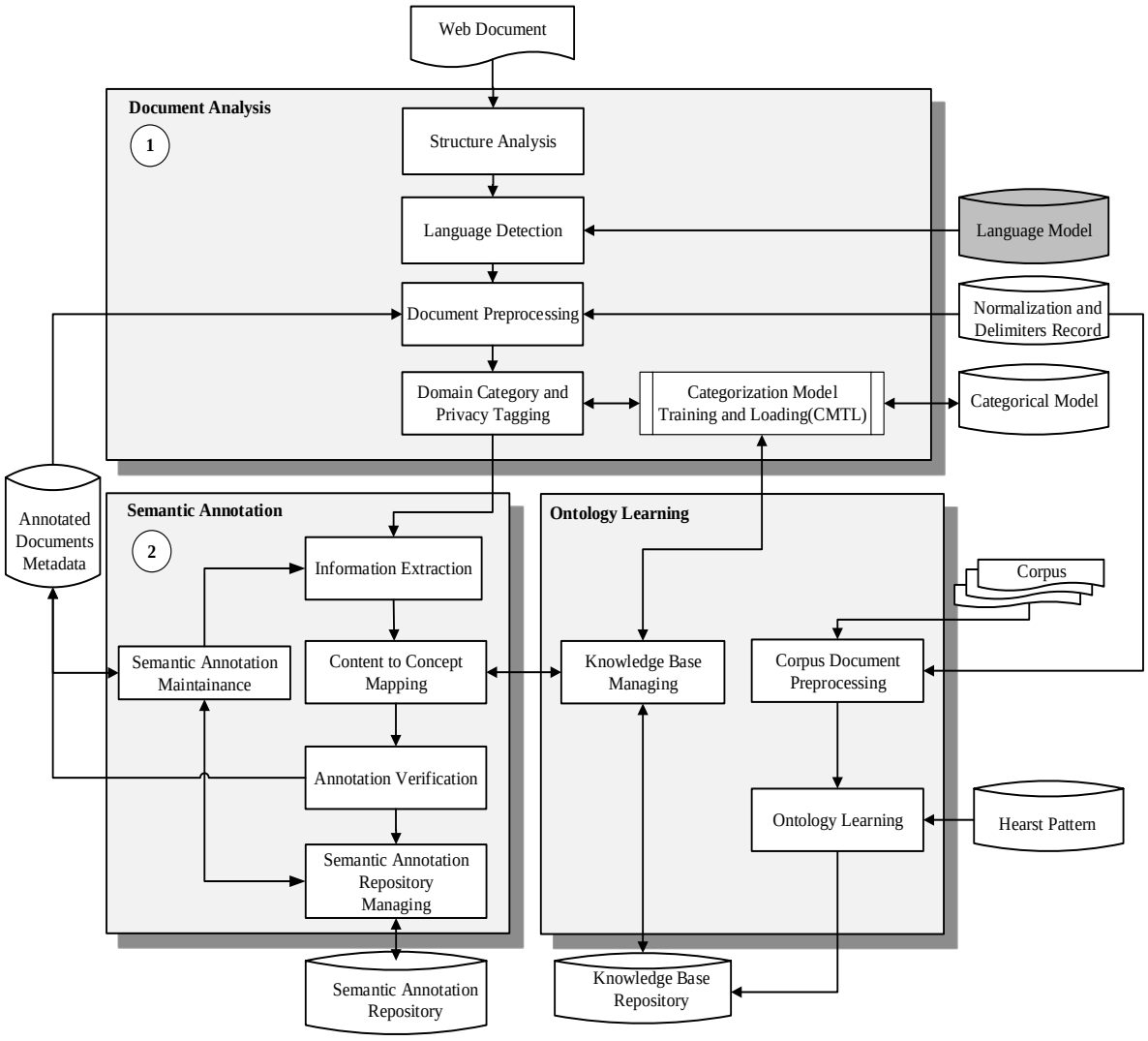

Figure 4.1: *Proposed Framework for Web Documents Semantic Annotation*

4.2 The Proposed Framework

This section presents details of the proposed framework for automatic semantic annotation of documents. As it can be seen from Figure 4.1, this proposed framework consists of a series of components which perform different transformations on an input document and finally stores its semantic information in a machine readable and understandable format that could be used by humans or software agents.

The framework employs three comprehensive set of processes named as, document analysis, ontology learning, and semantic annotation. In this framework, CMTL and ontology learning are required to be processed and modeled first, to use it as an input for domain categorization and content to concept mapping sub-processes respectively. Label 1 and 2 of Figure 4.1 are indicators of where the semantic annotation starts with document analysis (1) and continues with semantic annotation (2). Thus, this framework conducts document analysis before annotation to make the document suitable for upcoming components. After the document is analyzed, semantic annotation is conducted using ontology as a world/domain knowledge. If document semantic annotation is verified, semantic annotation repository manager stores the semantic annotation into the repository in a way that accessible and understandable to software agents and other semantic annotation users in need to understand document semantic information.

4.2.1 Document Analysis

Document analysis is a comprehensive process that provides document structure parsing, language identification, domain and privacy categorization, and document preprocessing. The details of all sub-processes are presented under the following subsections.

4.2.1.1 Structure Analysis and Language Detection

Structural analysis of the document is an essential task in document information gathering and dissemination to identify main structural elements (Line 3 of Algorithm 4.1) of the document. Sections such as authors, title, abstract, introduction, paragraphs, sentences, conclusion, and references are considered as the main structural elements used to fill the structural template that has elements shown in Table 2.1. As structure element detection, structure indicators such as position in a document, punctuation marks, and a newline are used with different document parsing methods. As a result, the filled template (Line 4 algorithm 4.1) provides information

regarding the structure of the document content which is used to understand how items are arranged with their information weight. For example, terms or concepts appeared in a title of the document have a higher information weight than that of appeared in the body section. Algorithm 4.1 shows how to extract structural elements from a given document and fill the structural template of the document.

Algorithm 4.1: *Structure Analysis Pseudocode*

```
Input: WD: Web document
       MST: Metadata and Structural Template
       SD: Structure Delimiters
Output: FDT: Filled Document Template
Begin:
1.  wd_format=detectFormat(WD)
2.  If wd_format is valid
3.      wd_segment = SegmentDocument(WD, wd_format, SD)
4.      FDT= FillTemplate(MST, wd_segment)
5.      Return FDT // structured document after "template filling"
6.      End If
7.  Else
8.      Return invalid_document
End
```

As language detection, LangID [111] is adopted, since it has better accuracy and also solved character n-gram based language detection challenges. As presented by Lui and Baldwin [111], LanID trained over a naive Bayes classifier with a multinomial event model, over a mixture of byte n-grams for 97 different languages provided in Annex B.

4.2.1.2 Document Preprocessing

Preprocessing is essential in this framework, because unstructured textual data are not normalized and absence of preprocessing limits further document processing and understanding. If there are also irrelevant and duplicated information within a document, discovering knowledge is difficult for upcoming document understanding and tagging components. Thus, this component is dedicated to preprocess web documents using redundancy checking (Algorithm 4.2), normalization (Algorithm 4.3), and splitting (using word and sentence delimiters found in normalization and delimiters record repository). The overall preprocessing workflow is illustrated in Figure 4.2.

Redundancy checking: Under preprocessing, redundancy checking is used to check the given document vector from the annotated document metadata repository (Line 2 and 3 of Algorithm 4.2), which is used to avoid redundant annotation of the same document, because one document should be annotated only once if there is no change on that document. This helps in avoiding computational and storage resource wastage due to document annotation redundancy. If redundancy flag is true (i.e., the given document is previously annotated), the document analysis and the overall semantic annotation are aborted, else the preprocessing proceeds into document normalization.

Algorithm 4.2: *Redundancy Checking Pseudocode*

```
Input: WD: Web document
       ADMR: Annotated Documents Metadata Repository
Output: RF: Redundancy Flag
Begin:
1.  WDVec= CreateVector(WD)
2.  VecRecord= ADMR.query(WDvec)
3.  If VecRecord not NULL
4.       RF=True
5.  Else
6.       RF= False
7.  Return RF
End
```

Normalization: If redundancy flag is false, redundant characters with the same sound and other expressions referring to the same content element have to be mapped into their normalized form to have common representations of characters and words. Thus, as shown in line 2 and 3 of Algorithm 4.3, this task transforms document content into a predefined standard format using normalization record stored in normalization and the delimiters record repository.

Algorithm 4.3: *Document Normalization Pseudocode*

```
Input: WD: Web document
Intermediate: CMD: Character Mapped Documents
Output: ND: Normalized Document
Begin:
1.  MR= LoadMappingRecords()
2.  CMD= CharacterMappings(WD, MR.characterMaps)
3.  ND= ExpressionMappings(CMD, MR.expressionMaps)
4.  Return ND
End
```

Therefore, this normalization record has to be created and stored in normalization and the delimiters record repository in order to produce a standard format previously chosen. Finally, splitting task is required through the use of writing system delimiters of the natural language for sentences, words, and paragraph detection as sub-elements of the given document for semantic annotation.

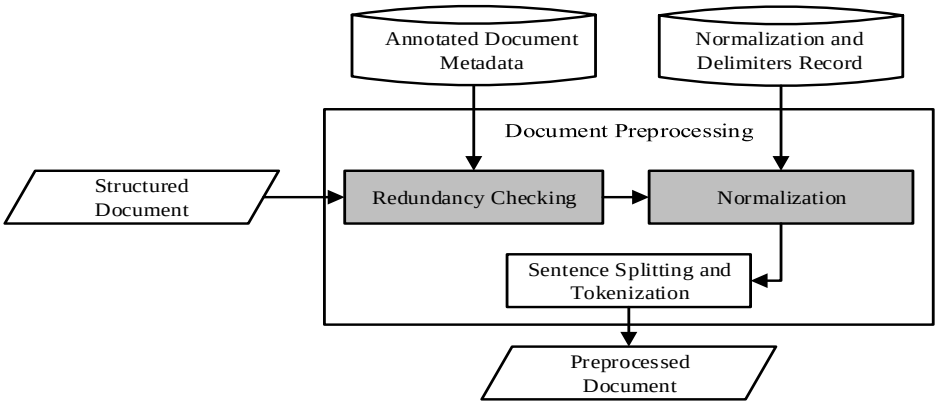

Figure 4.2: *Document Preprocessing Workflow*

4.2.1.3 Domain and Privacy Tagging

In this research, domain categorization and labeling aims to minimize the cost and ambiguity of instance searching at content to concept mapping phase of semantic annotation. The mapping cost is minimized by due to domain awareness that provides an option of directly searching ontology of the specified domain than looking over all domain ontologies based on the assumption that one document belongs to one domain.

Algorithm 4.4: *Domain Categorizer Modeling Pseudocode*

```
Input: TrainingDS: Domain labeled Dataset
Output: DCM= Domain Categorization Model
Begin:
1.  TokenizedTrainingDS = Tokenizer(TrainingDS)
2.  WeightedTokens = ComputeTFIDF(TokenizedTrainingDS)
3.  domain_priories=  compute_domain_priori(WeightedTokens)   /*
    using naïve Bayes priori probability computer */
4.  DCM = createModel(domain_priories, format)
5.  SaveModel(DCM)
End
```

As shown in Algorithm 4.4, the categorization model is proposed to be trained using naïve Bayes classifier that used for categorical modeling. As shown in line 2-3 of Algorithm 4.4

naïve Bayes used tokenized and TFIDF weighted dataset and computes the conditional dependency between the content and the domain. In this process, as a sub-process CMTL aims to train and load domain categorization model and it is also dedicated to request knowledge base manager for privacy gazetteers list. After the privacy gazetteers are returned from the knowledge base manager, the process proceeds into document privacy tagging that allows the owner of the resource (in this case document) to define, manage and enforce access conditions applicable to the resource and secures semantic annotation of sensitive documents.

Algorithm 4.5: *Knowledge Oriented Document Privacy Tagging Pseudocode*

```
Input: SD: Structured Document
    PTKs: Privacy Triggering Keywords(Gazetteers)
    TKw: Triggering Keywords Weight
    ODOP: Owner Document Privacy Preference
    PT: Privacy Threshold
    W2VModel: Word2Vec Model and UPP: User Privacy Period
Variable:
    Key: RSA Public and Private Key and PL: Privacy Lifetime
Output: PT: Privacy Tag, RSAKey: RSA Key
Begin:
1.  If ODOP is True
2.      Key=GenerateKey(RSA)
3.      PL= UPP
4.      End If
5.  Else
6.      SD_PrivacyTerms=PrivacyTermsExtractor(SD, PTKs, W2VModel)
7.      PrivacyWeight=ComputePrivacyWeight(SD_PrivacyTerms, TKw)
8.      If PrivacyWeight > PT
9.          key = GenerateKey(RSA)// RSA as key generation technique
10.         PL= 'Permanent'
11.         End If
12.     Else
13.         Key=NULL
14.         PL=NULL
15.         End Else
16.  End Else
End
```

As a sub-process, privacy tagger permits two specifications of privacy as owner oriented and knowledge oriented in a unified frame. The notion of knowledge-oriented privacy (Algorithm 4.5) is achieved by analyzing sensitiveness of a document content in terms of predefined

knowledge (i.e., a list of system suggested privacy keywords approved and weighted by domain experts). On the other hand, owner oriented privacy depends on owner preference regarding the sensitivity of the content and Line 1 of Algorithm 4.5 is used for checking document privacy preference of the owner. Thus, owner-oriented (Line 1-4 of Algorithm 4.5) and knowledge-oriented (Line 6-11 of Algorithm 4.5) privacy tagging are used for a given document as a way of adding point of trust for the owner of the document. Finally, lifetime of privacy has to be tagged either as temporal or permanent. The type of privacy is used as criteria to decide the lifetime. If privacy criterion is knowledge-oriented, lifetime set to permanent else document owner lifetime interest is used (Line 3 of Algorithm 4.5). If the lifetime is temporal, temporary procedure has to be created to clear the privacy when the lifetime of the privacy expires.

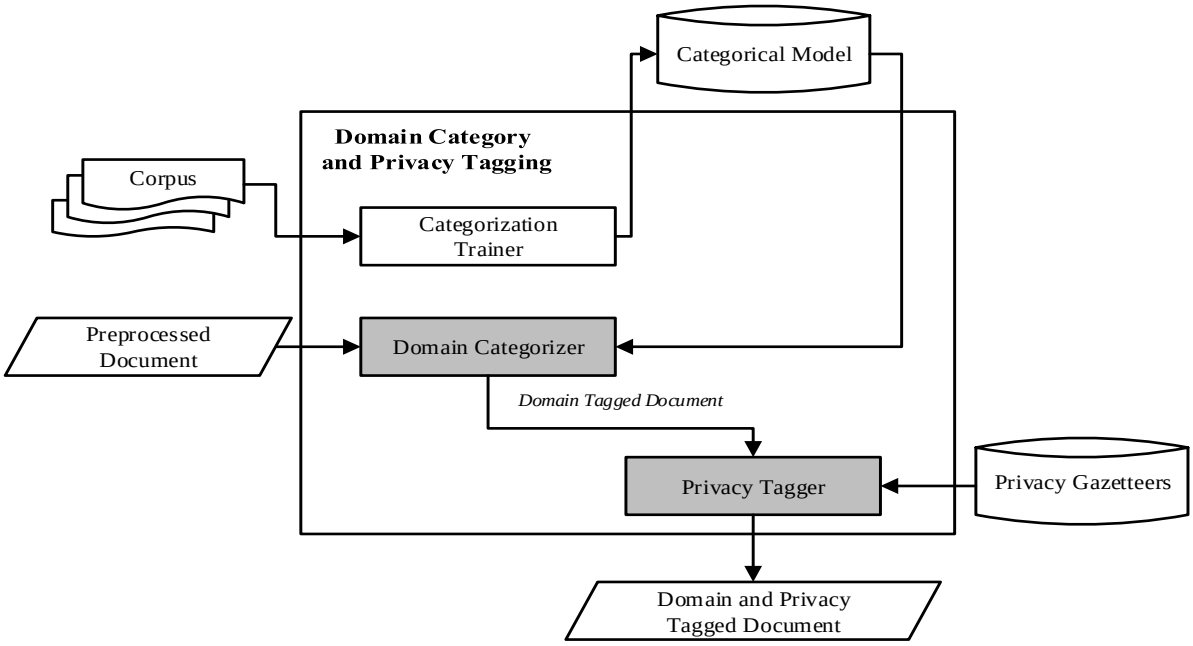

Figure 4.3: *Domain and Privacy Tagging Workflow*

4.2.2 Semantic Annotation

This section presents components working together to annotate documents semantically, first relevant content is extracted with information extractor (Line 1 of Algorithm 4.6) and content to concept mapper is dedicated with mapping extracted contents using premodeled ontology (Line 2 – 9 of Algorithm 4.6). Once the relevant document contents are mapped, annotation verifier (Line 10 of Algorithm 4.6) computes the score of mapping and invokes the repository manager for persistency if the mapping is verified for the given level of annotation. Detail of

each annotation sub-processes (Line 1-20 of Algorithm 4.6) are discussed under upcoming sections.

Algorithm 4.6: *Higher Level Semantic Annotation Pseudocode*

```
Input: AWD: Analyzed Web Document
    DOPP: Document Owner Privacy Preference
    LA: Level of annotation
Intermediate:
    RDC: Relevant Document Contents, LDO: Loaded Domain Ontology,
    Mappings: List of maps between content and concept
Variables:
    Mappings = []
Output: SA: Semantic Annotation
Begin:
1.  RDC = InformationExtractor(AWD)
2.  LDO = LoadOntology(AWD.domain)
3.  For each content in RDC
4.      concept=SearchOntology(content)
5.      If (concept != NULL)
6.          map=content.map(concept, linkweght)
7.          Mappings.add(map)
8.          End if
9.  End For
10. VFlag=AnnotationVerifier(Mappings, LA)
11. If VFlag=True
12.     SA=CreateRDF(Mappings)
13.     AddMetadata(AWD.vector, AWD.Metadata)// annotated document log
14.     If(AWD.Privacy = True)
15.         StorePrivateAnnotation(SA)
16.         End If
17.     Else
18.         StorePublicAnnotation(SA)
19. End If
End
```

4.2.2.1 Information Extraction

In IE, statistical techniques are not sufficient for text mining, thus, better document processing and information gathering could be performed in order to integrate IE technologies into the annotation in a way that it identifies entities in textual document which are instances of a particular concept with other relevant information. Thus, under this process, semantically relevant elements, such as entities (e.g., concepts and instances) are identified using NER.

Moreover, relevant sentence selection, and paragraph gloss tagging are also considered as additional processes of extracting information from web documents for better tagging. As shown in Figure 4.4, IE component can understand document at three levels of granularities named as term, sentence and paragraph.

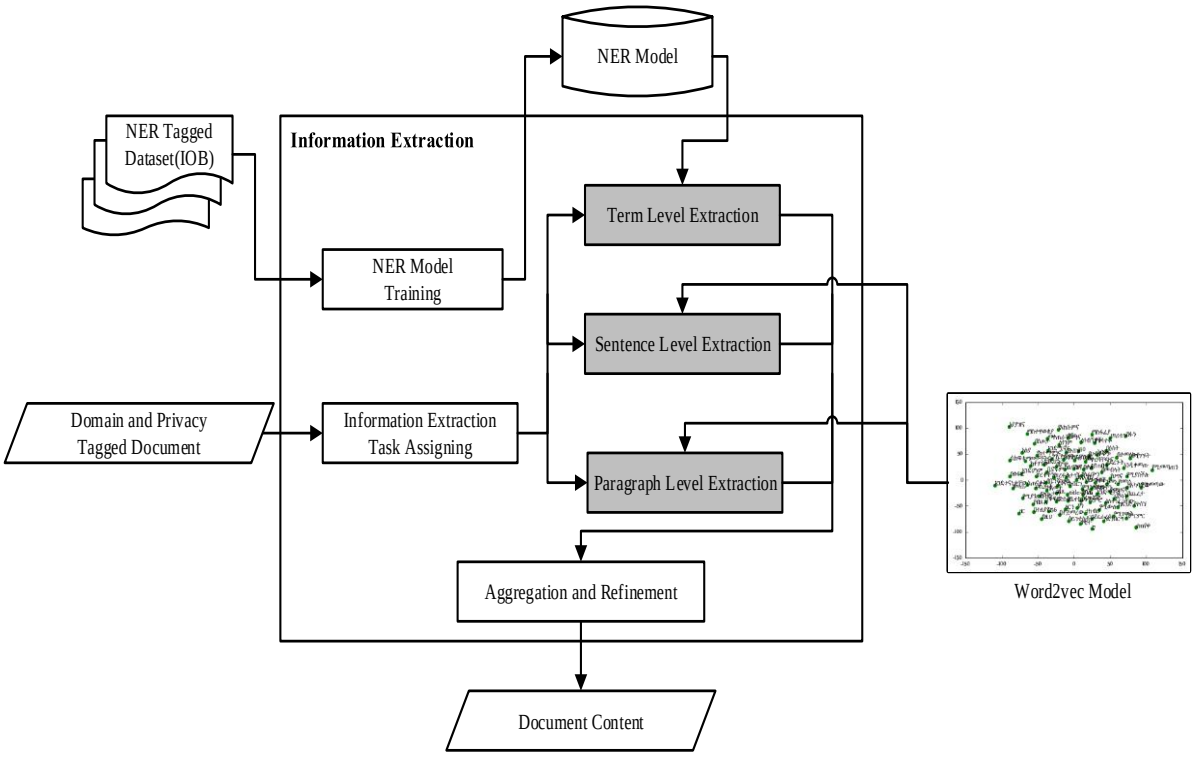

Figure 4.4: *Document Information Extraction Workflow*

i. Term Level Extraction

For term level extraction, NER is used to identify all expressions referring to a specific entity or object in the documents or texts, like names of locations, places, countries, and persons. NER is considered to be the basic task of IE and is one of the essential processes of NLP systems. The overall process is presented in Algorithm 4.7 in two phases as content extraction and refinement and NER interacting components are shown in Figure 4.5.

As NER is divided into two tasks, the first task is identifying terms in textual documents, and the second task is classifying the extracted terms (Line 6-9 of Algorithm 4.7) into predefined classes of interest, such as organization, location, and person as shown in Figure 4.6. According to Lample [39], token-level evidence for "*being a name*" used both morphological and distributional evidences.

Table 4.1: *NE Tagsets with Examples*

NE Tagset	*Description*	*Example(s)*
Person (PER)	Person, Nationals	Abebe Kebede, Ethiopian
Location(LOC)	Cities, Continents, Parks	Addis Ababa
Organization (ORG)	Institutions, Companies,	Ministry of Education
Organism (ORGM)	Animal, Plant, Virus,	Sheep, Tree
Climate(CMT)	Denotes the climate condition	Summer, Winter
Foods(FOOD)	Plant/Animal products	Butter, Bread
Disease(DS)	Disease affecting living things	Tuberculosis
Events(EV)	Conferences, Workshops, Meetings	Annual meeting
Date(DATE)	Denotes Dates, Months, Years	Monday
Time(TIME)	Denote Hour	Afternoon
Currency(CUR)	Denote Birr, Dollar,	Birr
Matter(M)	Denote matters	Car

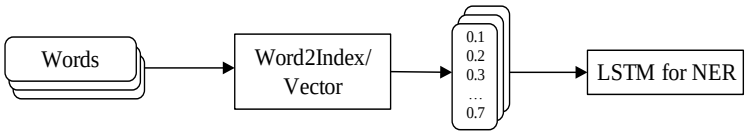

Figure 4.5: *LSTM Components Interaction for NER as Term Level Extraction*

By considering recent success of neural network techniques for various NLP and classification tasks, we adopted neural network technique [39]. The neural network is used to learn NE features from data with minimal handcrafted features and resources approved by linguists. Thus, for this work RNN (LSTM) based supervised information extraction that has components shown in Figure 4.5 is used for term level information extraction by considering the word as smallest linguistic unit for NER modeling.

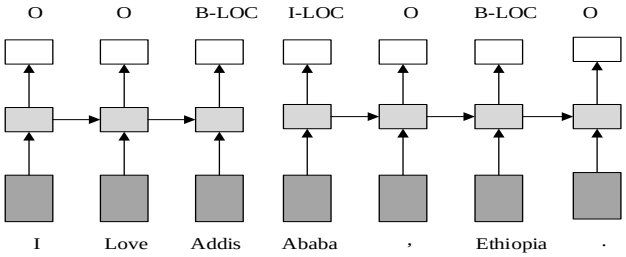

Figure 4.6: *Sequence Tagging for NER*

The adopted LSTM has the capability to learn from data with no language-specific resources or features beyond a small amount of supervised training data labeled with tag sets shown in Table 4.1 [38]. In this process, unlabeled corpora are used for word vector representation as depicted in Figure 4.5 in which word2vec is used as a word embedding component and RNN. In NER LSTM[21] is used for the classification purpose as a special kind of RNN with capability of learning long-term dependencies. As shown in Table 4.1, the listed commonly occurring NE tag sets are used to understand and extract set of entities from the document which is in the process to be annotated semantically.

Algorithm 4.7: *Term Level Extraction Pseudocode*

```
Input: SD: Structured Document
       D: Delimiters
       Word2index: word to indexer mapper
       LSTM_NERM: LSTM Named Entity Recognition Model
Output: TL: Term List
Begin:
1.  SDEntities = []
2.  SD_Sentences=SentenceSplitter(SD, D.sentenceDelimiter)
3.  NERM = load(LSTM_NERM)
4.  For each sent in SDSentences
5.      sent_tokens = Tokenizer(sent, D.tokenDelimiter)
6.      indexed_sents = SentenceIndexer(sent_tokens, word2id)
7.      entities_index=ExtractEntities(indexed_sents, NERM)
8.      For each index in entities_index
9.          If index ≠ 0 // index 0 indicates unknown token
10.             SD_Entities.add(Sent_Entities)
11.             End If
12.     End For
13. End For
14. TL=Refine_Entities(SD_Entities) /* refine entities by checking
    less relevant and overlapping entities */
15. Return TL
End
```

Once NER is modeled, named entities are extracted from the document as shown in Algorithm 4.7. In this process of term level extraction, first document has to be splitted into a sequence of sentence (Line 5 of Algorithm 4.7) and each sentence in splitted sentences set have to be

[21] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

converted into a sequence of indexes/vectors (Line 6 of Algorithm 4.7) to be processed by LSTM based NER. Next, each sequence of indexes/vectors is tagged with appropriate NE tag indexes provided in Annex E using the loaded NER model (Line 7 of Algorithm 4.7). After each sentence are tagged with NE indexes, each element of the sentence tagged with indexes different from zero can be considered as named entities and added into the document named entities list (Line 8-12 of Algorithm 4.7). Finally, as shown in line 14 of Algorithm 4.7, to refine the result of NER, overlapping entities and less relevant entities are checked and the list is refined.

ii. Sentence Level Extraction

This level of document understanding and information extraction is used for abstract and (or) conclusion sections of the document. If the document has no abstract and conclusion section (Line 3 of Algorithm 4.8), the sentence level document tagging can be empty node, else the score value is computed in terms of feature vector and selects the one with maximum information content using feature vectors to benchmark vector similarity computation as shown in line 5-9 of Algorithm 4.8. As sentence level tagging, pseudocode depicted in Algorithm 4.8 starts with an information content comparison (i.e., between abstract and conclusion) and end with final representative sentence's triplet representation (Line 18 and 19 of Algorithm 4.8). The winner of abstract and conclusion is passed to representative sentence selection with the help of feature vector and three basic terms are extracted from the representative sentence using their POS tag and position in a sentence as subject (noun), predicate (verb), and object (noun) (Line 18 of Algorithm 4.8).

Algorithm 4.8: *Sentence Level Extraction Pseudocode*

```
Input: A: Abstract, C: Conclusion
       SB: Similarity Benchmark // unit vector of one (1's)
       W2VModel: Word2VecModel
Intermediate:
       MRS: More Relevant Section
       SRV: Sentences with Relevancy value
       RSent: Representative Sentence
Output: Triplet: Document Representative Sentence as triplet
Begin:
1.  If A ≠ NULL and C==NULL then MRS=A and go to Line 10
2.  Else if A==NULL and C ≠ NULL then MRS=C and go to Line 10
3.  Else if A==NULL and C==NULL then Triplet=NULL and go to Line17
```

```
4.  Else
5.      Av =ComputeAverageVector(A, W2VModel) //Convert abstract
        into vector
6.      Cv=ComputeAverageVector(C,W2VModel)//Convert conclusion
        into vector
7.      Ar =Sim(Av, SB)// Abstract section relevancy
8.      Cr =Sim(Cv, SB) // Conclusion Section relevancy
9.      If(Ar > Cr) then MRS=A else MRS=C
10.     End Else
11. MRS_sentences=SentenceSplitter(MRS)
12. For each sent in MRS_sentences
13.     sent_vec= ComputeAverageVector(sent, W2VModel)
14.     sent_relevancy=Sim(Sent_vec, SB)
15.     SRV.add(sent, sent_relevancy)
16. End For
17. RSent=RepSentExtractor(SRV)/* sent with higher relevancy*/
18. Triplet = TripletExtractor(Rsent)/*POS and position is used */
19. Return Triplet
End
```

iii. Paragraph Level Extraction and Aggregation

Our paragraph level extraction is to add document information at the paragraph level by converting each paragraph into semantic vector. Paragraph vector logic proposed by Dai *et al.* [112] is adopted to create representations of paragraphs as vector in unsupervised fashion. The Dai *et al.* [112] paragraph vector is an extension to the word2vec as paragraph2vec. Thus, we used paragraph vector logic to represent information about the paragraph which act as a memory of the topic of the given document paragraph.

Table 4.2: *Extracted Content Refinement Metrics*

Refinement Metrics	***Description(s)***
Redundant token checking	Redundant content has to be removed (by checking redundant tokens from set of contents)
Content Normalization	Character Mapping (as discussed under document preprocessing normalization)
Tag content token with stem	To avoid content with the same morphological reflection and inflections

Accordingly, paragraph2vec, an unsupervised extended version of word2vec approach is used to learn fixed-length feature representations of document paragraphs. Once the feature vector is computed, each paragraph is tagged with paragraph's *is_about* which is processed (Line 3-5 of Algorithm 4.9) by querying the word2vec model of the domain using the computed vector. Finally, the result of the three granularity level (term, sentence, and paragraph) is aggregated and refined using refinement metrics stated in Table 4.2.

Algorithm 4.9: *Paragraph Level Tagging Pseudocode*

```
Input: PL: Paragraph List and
       W2VModel: Word2VecModel
Intermediate:
       PLV: Paragraph List Vector
Output: PIAL: Paragraph Is_About List
Begin:
1.  For each paragraph p in PL
2.       P_vec=ComputeAverageVector(p, W2VModel)
3.       top_similar_term=W2VModel.most_similar(p_vec, topn=1)
4.       p_isabout= p.isAbout(top_similar_term)
5.       PIAL.add(Map(p, p_isabout)
6.  End for
7.  Return PIAL
End
```

4.2.2.2 Content to Concept Mapping

This process is responsible for mapping document terms and information extracted at the phase of information extraction with ontological concepts as shown in Figure 4.7. Most of the approaches used so far for semantic annotation are generally based on term extraction and extraction of relations between these terms. However, in most cases the context of these terms are ignored, and that is why semantic relatedness computation proposed in [113] is adopted with modification to suit into our work as shown in Algorithm 4.10. This algorithm has two-phases used for mapping named as searching phase (1-11 of Algorithm 4.10) and suggestion phase (12-18 of Algorithm 4.10).

Algorithm 4.10: *Content to Concept Mapping Pseudocode*

```
Input: TL: Term List
       DO: Domain Ontology
       SD: Structured Document
       CW: Context Window // Default window=4
```

```
    ST: Similarity Threshold //Default is 0.5
    W2VModel: Word2Vec Model
Variable:
    Link_type = "Direct"
    LW= Link Weight // Default is 1 for direct link
Output: ML: Mapped List, PIAL: Paragraph's Is_About List
Begin:
1.  For each term in TL
2.      term_concept =SearchTerm(term, DO)/* use lexical and stem
                                              based checking */
3.      If term_concept = NULL
4.          cand_concept=SearchSimilarConcept(term, DO, W2VModel)
5.          sim_t_c =similarity(term, cand_concept)
6.          If sim_t_c > ST // using word2vec
7.              term_concept= cand_concept
8.              Link_type = 'Similarity'
9.              LW = sim_t_c
10.             End If
11.         Else
12.             term_context= ContextExtractor(term, SD, CW)
13.             instance=GenerateInstance(term_context)
14.             DO.suggest(instance)
15.             term_concept=instance
16.             Link_type = 'Suggestion'
17.             LW=0
18.             End Else
19.     End If
20.     ML.add(Map(term, term_concept),Link_type,LW) /* to add
                content to concept map in to mapping list with
                their link tyoe and link weight */
21. End for
22. Return ML
End
```

In this mapping process, searching phase comprises three consecutive steps: lexical searching, stem based searching, and similarity (embedding) based searching. As the second phase of mapping (line 11-18 of Algorithm 4.10), if concept matching is failed, instance suggestion is conducted to suggest new instance for ontology population to domain ontology engineer (Line 14 of Algorithm 4.10) and he/she will accept or reject the suggested content in terms of its relevancies for the domain knowledge base.

4.2.2.3 Annotation Verification

Once concept to content mapping is done, to verify annotation mappings validity, the verifier is dedicated to mappings score computation using the proposed Equation (1). Thus, this component is used for verification of the annotation before committing the overall process. The process of verifying semantic annotation is presented in Algorithm 4.11 and it starts with annotation score computation (using scoring metrics under Table 4.3) and ends up with returning verification flag. The verification criteria, level of annotation has to be given by the document owners as *simple*, *moderate*, or *strict* and the computed score is compared with the corresponding annotation level threshold. If the annotation level preference is simple, less than 50% annotation score is acceptable, but for moderate and strict, annotation score has to be greater than 50% and near 100% respectively. If the owner is not setting the preference for the level of annotation, the default level of annotation, *moderate* level is used as a level of annotation for verification.

Algorithm 4.11: *Semantic Annotation Verification Pseudocode*

```
Input: CCML: Content to Concept Map List
    LA: Level of Annotation{Simple: S, Moderate: M, Strict: St}
    LT: Level Threshold{Simplet:0.0, Moderatet:0.5, Strictt:0.9}
Intermediate:
    CIW: Content Information Weight
    CTL: Concept Term List
    ACW: Average Concept Weight
    MW, MWL: Map Weight, Map Weight List
    NMWL, ANMWL: Normalized Map Weight List, Average NMSL
Output: AVF: Annotation Verification Flag
Begin:
1.  For each map m in TCML
2.      CW =0
3.      CIW=ComputeInformationWeight(m.content) //using TFIDF
4.      CTL=ConceptToTermMapping(m.concept)
5.      For each term t in CTL
6.          tw=ComputeInformationWeight(t)
7.          CW = CW + tw
8.      End for
9.      ACW= CW/size(CTL)
10.     MW=(CIW+ACW)*m.link_weight
11.     MWL.add(m,MW)
12. End for
13. NMWL = NormalizeMWL(MWL) // Normalize each value into 0-1 range
```

14. ANMWL = sum(NMWL)/size(m)
15. If (LA = 'Strict' and ANMWL > $Strict_t$)
16. AVF = True
17. End If
18. else if (LA = 'Moderate' and ANMWL > $Moderate_t$)
19. AVF = True
20. End else
21. else if (LA = 'Simple')
22. AVF = True
23. End Else
24. else
25. AVF = False
26. End Else
27. If AVF = True
28. SToreWDVec&Metadata(WD_{vec}, WDM)
29. End If
30. Return AVF
End

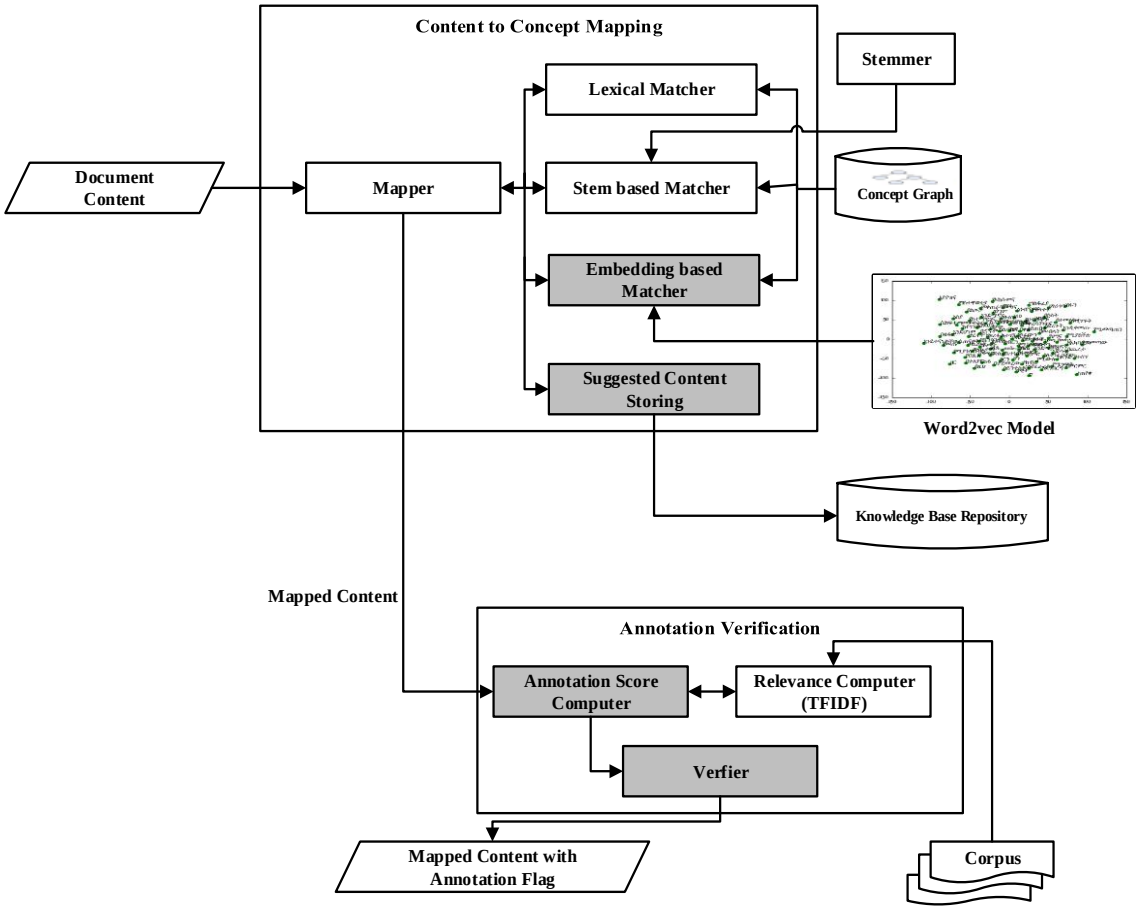

Figure 4.7: *Content to Concept Mapping and Verification Workflow*

After comparing mapping score with preference threshold, the verification flag is set to true (for a score greater than preference threshold) or false (for a score less than or equals to preference threshold) as shown in line 15-26 of Algorithm 4.11. If the flag is true, the verifier store document metadata into the annotated document repository (Line 28 of Algorithm 4.4) and the annotation process proceeds into semantic annotation storage, otherwise if the verification flag is false annotation procedure is aborted due to unverified semantic annotation.

Table 4.3: *Verification Scoring Metrics*

Scoring Metrics	***Description***
TFIDF	Content and Concept TFIDF
Ontology Lookup Type	Direct lexical and stem based lookup type have weight of 1, similarity based lookup has content to concept cosine similarity as a weight, and suggestion has 0 mapping weight

$$\mathbf{FK\_SA_{score}} = \frac{\mathbf{1}}{\mathbf{K}} \sum_{\mathbf{i=1}}^{\mathbf{K}} \mathbf{n\_ms_{C2C_i}} \quad (1)$$

Where:

- FK_SA_{score} is semantic annotation score,
- K is total number of maps, and
- For given mapping score list (msl), each map score (ms) has to be converted into value between 0 and 1 using:

$$n\_ms_{c2c} = \frac{ms_{c2c} - msl_{min}}{msl_{max} - msl_{min}}$$

4.2.2.4 Semantic Repository Managing

The last issue in semantic annotation goes to how to store semantic information of a given document in a form which is understandable and processable to the machine and reusable later by other software agents for meaning based applications. Thus, this component is responsible for the persistence of semantic annotations made by previous components. In this process, first verification flag is checked (Line 1 of Algorithm 4.12), and if the flag is true, mappings,

representative sentence triplets and paragraphs *is_about* list are serialized into RDF format and stored in semantic annotation repository. We used RDF for storage due to its capability to make statements about any resource, and it is also recommended by W3C for representation of semantic annotation. As shown from line 6 of Algorithm 4.12, the storage might be signed with privacy data if the verification flag is true.

Algorithm 4.12: *Semantic Annotation Persistency Pseudocode*

```
Input: ML: Mapping List
       VF: Verification Flag
       PD: Privacy Data
       PIAL: Paragraph Is_About List
       RST: Representative Sentence Triplet
       WDvec: Web Document Vector
       WDM: Web Document Metadata
Output: PM: Persistency Message
       RDFL: RDF List
Begin:
1.  If VF = TRUE // If the annotation is verified
2.       RDFL = CreateRDF(ML, PIAL, RST)
3.       For each rdf in RDFL:
4.            StoreRDF(rdf)
5.       End for
6.       If PD.flag=True then TagPrivacy(PD.key)
7.   End If
End
```

4.2.2.5 Semantic Annotation Maintenance

In a realistic semantic web scenario, incoming information that has to be annotated does not require only annotation, but also continuous adaptation to document evolution. If the document owner notified that his/her document is evolved or periodic change detection flag is set as *'true'* (Line 1 of Algorithm 4.13), semantic information of document has to be maintained. For reannotation, first similarity has to be computed between the original document vector found in annotated document metadata repository and the one reported or detected as evolved document (Line 2-4 of Algorithm 4.13). Thus, as shown in line 5-9 of Algorithm 4.13, the document is reannotated if the computed similarity value is different from 1 (as indicator of changes in a document that leads into reannotation) in a way that helps to keep consistency between a document and its semantic annotation. In line with this, ontological change also has to be maintained by checking and modifying the ontological

concepts in order to reflect changes in the domain of interest, or to correct design flaws. This issue makes the task of understanding and detecting inconsistencies in ontology vital for ontology dependent applications such as semantic annotation.

Algorithm 4.13: *Semantic Annotation Maintenance Pseudocode*

```
Input: DS: Document States
    WD_old_vec: Vector of Web Document annotated as last version
    WD_new: New Version of Web Document //Accessed using a URI
    OA: Old Annotation
    UCRF: User Change Request Flag
    CSF: Change Scanning Flag
Intermediate:
    DSim: Documents Similarity
    TL: Term List
    RS: Representative Sentence
    PIAL: Paragraph Is_About List
Output: MDA: Maintained Document Annotation
Begin:
1.  If UCRF = True or CSF =True
2.      WD_metadata = getMetadata(WD)
3.      WD_new_vec = ComputeDocVector(WD_now)
4.      DS = Similarity(WD_old_vec, WD_new_vec)
5.      If DS != 1
6.          TL, PIAL, RS =ContentExtraction(WD_new)// Section 4.2.2.1
7.          Mapping=ContentToConceptMapper(TL) // Algorithm 4.10
8.          End If
9.      UpdateAnnotation(Mapping,WD_metadata,PIALS)// update and Tag
        date of maintenance
10.     End If
End
```

4.2.3 Ontology Learning

The six learning sub-processes of adapted bottom-up ontology learning approach [89] are shown in Figure 4.8 that start with corpus document preprocessing and concept extraction for the final concept graph generation. Thus, our ontology learner has six main sub-components named as: preprocessing, concept extraction, concept to domain mapping, concept pair extraction, taxonomic extraction and non-taxonomic relation extraction. Figure 4.9 shows an illustration of an ontology as an end result of ontology learning pass through phases depicted in Figure 4.8 that adapted from Lim *et al.* [89] ontology learning architecture. The acquisition

methods are statistical and neural networks in which conceptualization process transforms knowledge within a text into a machine-processable and understandable concept graph.

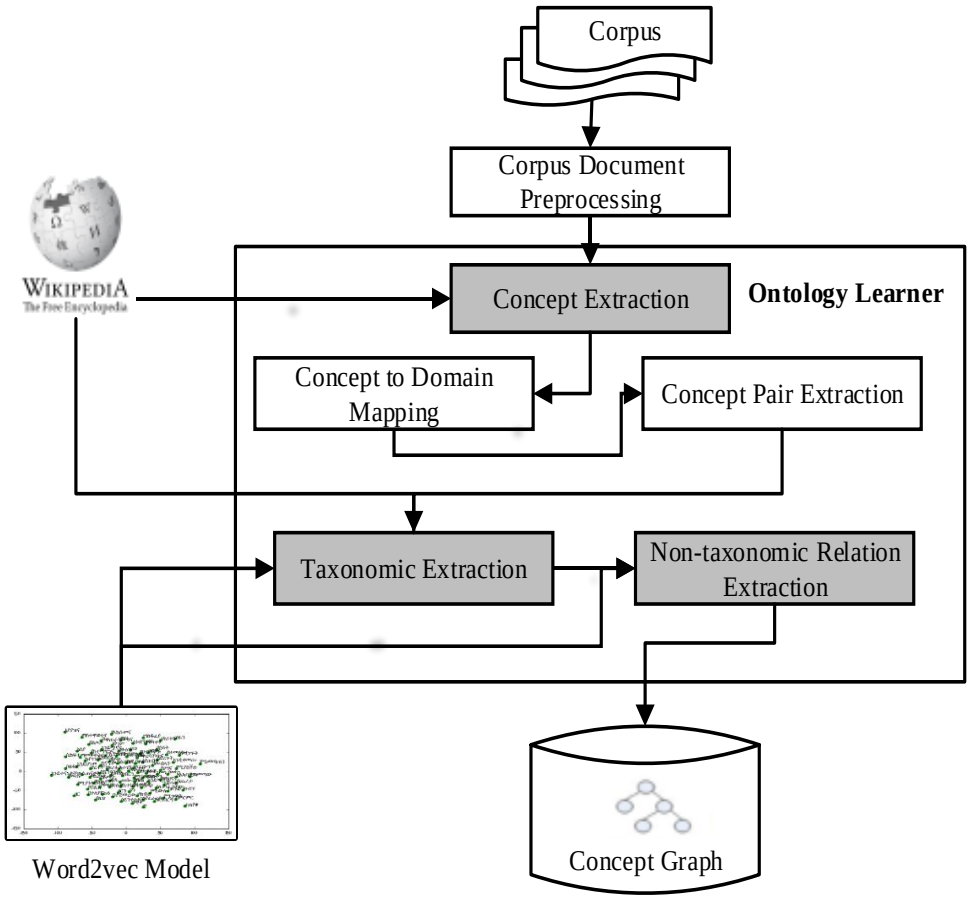

Figure 4.8: *Ontology Learning Architecture*

i. Corpus Document Preprocessing

As deep understanding usually decreases the speed of ontology learning, most of existing ontology learners use shallow text processing techniques such as tokenization, and POS tagging, to extract essential structures from input texts as shown in Figure 4.10. For this work, two categories of corpus are required, as one is labeled corpus and the other one is unlabeled corpus, the labeled one is used for concept to domain relationship mapping, while the unlabeled one is used for concept pair extraction and other phases of ontology learning. Both corpora must be relied on several preprocessing steps to make suitable for ontology learning by cleaning and preparing the corpus using different document preprocessing techniques such as sentence splitting, tokenization, normalization, and POS tagging.

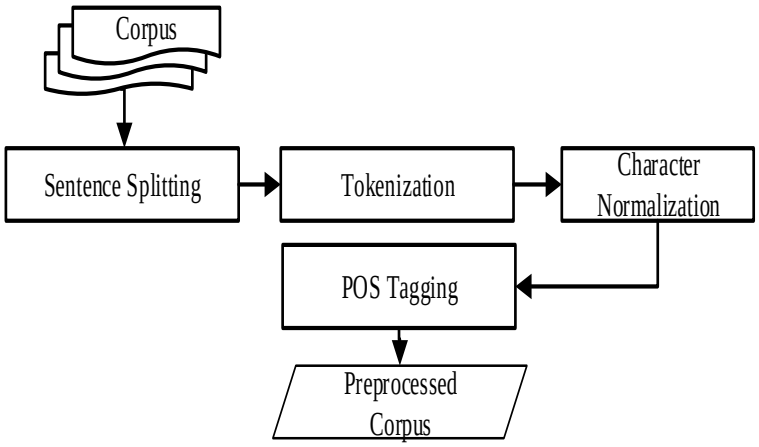

Figure 4.9: *Corpus Preprocessing Workflow*

ii. Concept Extraction

Nowadays, concept extraction from textual corpus is a very active area of research and several researches have been conducted in this area. Among those works [89] is the one, but our approach starts from POS tagged corpus at preprocessing phase rather than using dictionaries to help concept extraction process and recognize meaningful terms. The extraction process is based on the POS tag of each token and mapping to the online free encyclopedia, Wikipedia, for concept identification as presented in Algorithm 4.14. The concept extraction starts with preprocessed and POS tagged corpus and filter only nouns as candidate concepts (Line 1-6 of Algorithm 4.14) following the approach and findings discussed in [89]. Figure 4.11 shows candidate terms extraction workflow that starts with term extraction and ends with Wikipedia based extraction refinement.

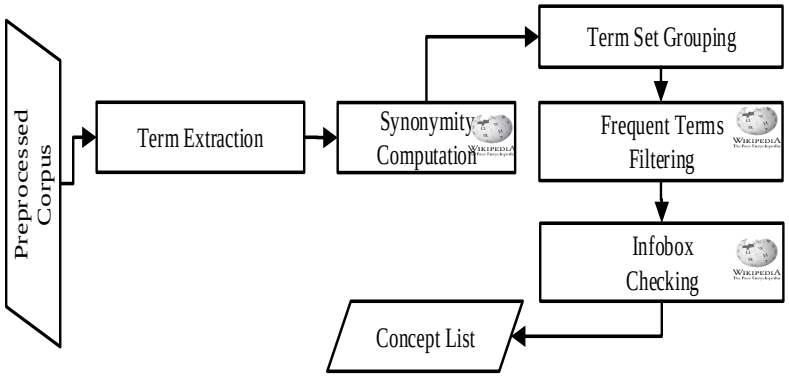

Figure 4.10: *Concept Extraction Workflow*

Under concept extraction, TFIDF is used as a relevancy weighting scheme to extract the top relevant domain terms using term frequency and inverse document frequency, which can be normalized as follows, to normalize term frequency divide the term frequency of a given term with total number of terms in a given domain and inverse document frequency could be normalized using most common approach of IDF normalization by Lo *et al.* [114] (i.e., respect to the number of domains not containing the term, $D - D_t$, and adding a constant value of 0.5 to both numerator and denominator to moderate extreme values) as defined in Equation (2) [114].

$$\mathrm{IDF}_{norm} = \log\left(\frac{(D-D_t)+0.5}{D_t+0.5}\right) \quad (2)$$

Where:

- t is a term
- D is total number of domains,
- D_t is number of domains containing term t, and
- 0.5 is extreme values moderator.

Algorithm 4.14: *Concept Extraction Pseudocode*

```
Input: C, WD: POS Tagged Corpus, Wikipedia dump
       D: Delimiters
       St, MWEt: Synonymity Threshold, MWE Threshold
       MF: Minimum Frequency
       VNW: Valid Number of Words for Multi Word Concept
Intermediate:
       TSent: Tokenized Sentence, NgramSet: N-gram Set,
       SWTSet: Single Word Term Set,
       MWTSet: Multi Word Term Set and TSet: Term Set
Output: LC: List of Concepts
Begin:
1.   For each sentence s in corpus C
2.       TSent = Tokenize (s, D.tokenDelimiter)//returns set of tokens
3.       For each token t in TSent
4.           If POS tag of t is Noun then SWTSet.add(t)
5.       End for
6.   End for
7.   For n = 2 to VNW // generate multi-words
8.       MWTSet = C.ExtractNgrams(n, TSent)
9.   End for
10.  TSet=SWTSet U MWTSet
11.  For each term ti in TSet
```

```
12.     tiG=extractglossfromWikipedia(ti)
13.     For each term tj ≠ ti in TSet
14.       tjG=extractglossfromWikipedia(tj)
15.       TGSim= similarity(tiG, tjG)
16.       If TGSim > St then SynSet.add(ti,tj)
17.     End for
18.   End for
19.   SynSettedTSet= Group(SynSet, TSet)
20.   SynSettedFreqTSet=SynSettedTSet.FilterFreqTerms(MF, WD)
21.   For each term t in SynSettedFreqTSet
22.    InfoBoxFlag=CheckInfoBox(t, WD)
23.    If InfoBoxFlag is True then LC.add(t)
24.   End for
End
```

To choose a relevancy threshold, terms have to be sorted based on their TFIDF values and once the term is sorted, to choose the thresholding term, the TFIDF variation of the two consecutive term is used as a parameter. Thus, term t_n with $w(t_n)$ relevancy weight would be considered as the thresholding term if the domain relevancy weight of t_n is very low in comparison to $t_{n-1,}$ i.e., t_n is much less relevant to the domain and any terms with relevancy weight less than or equal to $w(t_n)$ could be considered as less relevant terms of a given domain, and added to less relevant domain terms list.

In line with this, multi word expression (MWE) has drawn much attention in the field of ontology learning and review of 80 MWEs in [115] reported that POS sequences and other approaches relying on word statistical information were used for MWE. We adopted n-gram based MWE, a generic data-driven approach used as multi word concept extraction method and each n-gram score is computed using $\chi 2$ value. In this method, running the bigram n-times consecutively forms simple phrases with more than '*n*' words, (e.g., if we run twice, the first pass merges two words above the threshold into phrases, the second pass merges the phrases with other phrases and words) as shown in line 7- 9 of Algorithm 4.14.

The concept extractor builds sets of synonyms to help the matcher find terms that correspond to the same real world entities. A synonymity is between words having the same or nearly the same meaning as another word or those sharing (a part of) their semantics. Thus, without knowledge of these and other forms of synonymity, it is impossible to have good terms grouping. In this work, with an assumption that definition (gloss) of term would adequately

capture conceptual information, we used Wikification as automatic gloss extraction mechanism by querying Wikipedia and extracting definition part from Wikipedia articles for each extracted term.

The difficulty of synonymity extraction is the fact that even professional lexicographers always not agreed on whether two words are synonyms or not for evaluating a synonym extractor. Here, the main challenge is gathering word sequences that refer to the same meanings, because each word sequence has multiple meanings. Therefore, if terms share some of their semantics, it could be that they are referring to the same concept since ontologies are concerned with concepts and not with terms, this is a crucial step. Thus, term glosses similarity based synonymity detection is proposed in which pairs exceeded the synonymity threshold of gloss similarity are grouped as synonym term and the group is labeled with frequently occurring term in Wikipedia (Line 11-19 of Algorithm 4.14). In addition to synonymity, even though non taxonomic relationships are used as a common and important relationship in ontologies, correlated relationships, such as *has-website* and *has-data-about* is also another important information related to domain concepts.

In this process, Wikipedia based term extraction refinement for the extracted domain terms can be used in a way that provides significant refinement for the term extraction module for checking the term's popularity (using the frequency of terms in Wikipedia) and chance of being a candidate (using the existence of the term in infobox section). Therefore, as the last process in concept identification, Wikipedia based frequent term filtering, and subject-attribute-value, Wikipedia *infobox* section, checking are used as criteria for concept identification (Line 20-24 of Algorithm 4.14).

iii. Concept to Domain Mapping

Concepts extracted using concept extractor have no relationship rather than gloss based synonymity, their most representative domains are unknown. Algorithm 4.15 adopted from [89] associates a concept into the most representative domain identified with a domain dependency value computed using $\chi 2$ and R values. In Algorithm 4.15, Line 5, $R_{c,d}$ is computed to measure dependency value between concept *c* and domain *d* and the algorithm build vector contains domain contingency table into domain dependency list if $R_{c,d}$ value of Equation (5) [89] is beyond the dependency threshold value (Line 6-8 of Algorithm 4.15).

Line 9 returns the best domain that defines the concept by extracting the top value from the list and the concept is associated to the best domain (Line 10 of Algorithm 415).

Algorithm 4.15: *Concept to Domain Mapping Pseudocode*

```
Input: C: Set of Concepts, D: Set of Domains,
    Rt: Dependency Ratio Threshold and
    BLC: Balanced Labeled Corpus for each Domain
Variable:
    MRD: Most Representative Domain
Output: Associated Concepts to Domains
Begin:
1.  For each concept c in C
2.      DomainDependencyList= [ ]
3.      For each domain d in D
4.          χ²c_d = Computeχ²Value(BLC, c, d)
5.          Rc_d = ComputeRValue(BLC, c, d)
6.          If Rc_d > Rt then:
7.              DomainDependencyList.add(c, d, χ²c_d, Rc_d)
8.      End for
9.      MRD=DomainDependencyList.Top()
10.     Associate(c, MRD)
11. End for
End
```

We followed the principle of coherence in linguistics [47], hypothesis of relation, concepts which are semantically related tend to be "*near each other*" for concept to domain association. Table 4.4 shows contingency table, which is used as a first step in computing $\chi 2$ and R to measure candidate concept to domain dependency. To fill the 2x2 contingency table shown in Table 4.4, Observed Frequency ($O_{i,\ j}$) is computed using the co-occurrence frequencies between every concept c and domain d.

Table 4.4: *Candidate Concept to Domain Contingency Table*

	$\boldsymbol{d}$	$\neg\boldsymbol{d}$	$\sum$
$\boldsymbol{c}$	Oc,d	$Oc,\neg d$	$Oc,d\ +\ Oc,\neg d$
$\neg\boldsymbol{c}$	$O\neg c,d$	$O\neg c,\neg d$	$O\neg c,d\ +\ O\neg c,\neg d$
$\sum$	$Oc,d\ +\ O\neg c,d$	$Oc,\neg d\ +\ O\neg c,\neg d$	$Oc,d\ +\ Oc,\neg d\ +$ $O\neg c,d\ +\ O\neg c,\neg d\ = N$

These statistical measurements are employed to compute the dependency between the candidate concept and domain. The Expected frequency, Chi-square, and R values are defined in Equations taken from [89], (3), (4) and (5) respectively using $O_{i,j}$ from the 2x2 contingency table. The observed frequency compared to the expected frequency $E_{i,j}$ where $i \in \{c, \neg c\}$ and $j \in \{d, \neg d\}$. $E_{i,j}$ and χ^2 for concept c and domain d are defined in Equations (3) and (4) respectively.

$$E_{i,j} = \frac{\sum_{a \in \{c, \neg c\}} O_{a,j} \sum_{b \in \{d, \neg d\}} O_{i,b}}{N} \quad (3)$$

Where:

- χ^2 and R statistical measurement is employed in order to compute the dependency between the candidate concept and domain, and
- $O_{i,j}$ is observed frequency.

$$\chi^2 = \sum_{ij} \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \quad (4)$$

Where:

- O_{ij} denotes the observed frequency and
- E_{ij} denotes the expected frequency of ontological concepts c_i and c_j.

According to Lim *et al.* [89], the problem of using χ^2 statistic measurement is that it can measure the term dependency on a domain, but cannot measure whether the dependency is positive or negative, to solve this problem, measurement of a term dependency on a domain has to be computed using R value. R equations given by Lim *et al.* [89] are defined in Equations (5) and (6) as a ratio between observed frequency and expected frequency.

$$R_{c,d} = \frac{O_{c,d}}{E_{c,d}} \quad (5)$$

$$R_{c,d} = \frac{p(c,d)\, p(\neg c, \neg d) - p(c, \neg d)\, p(\neg c, d)}{p(c)\, p(d)} + 1 \quad (6)$$

Where:

- $R_{c,d}$ is the ratio between $O_{c,d}$ and $E_{c,d}$,
- $O_{c,d}$ is observed frequency for concept c in domain d,

- $E_{c,d}$ is an expected frequency for concept c in domain d,
- $R_{c,d} > 1$ positive dependency,
- $R_{c,d} = 1$ no dependency, and
- $R_{c,d} < 1$ negative dependency.

iv. Concept Pair Extraction

Once the extracted concepts are mapped into their corresponding domain the next phase in ontology learner is finding concept to concept relationship and exploiting dependency relations using the χ^2 and R measurements. Under this phase, the chi-square test aims at extraction of triples (c_1, c_2, d) in which d is a real value that indicates the correlation between two concepts as defined in Equation (4). There have been many attempts to determine similar concept pairs from text corpora and one of the widely used approaches in similarity computation is based on the distributional hypothesis [116], if terms occur in a similar context, then they tend to have similar meanings. The equation for χ^2 statistics is modified to measure the dependency between two concept c_1 and c_2, instead of between a concept and a domain used for the previous phase of concept-to-domain mapping. The co-occurrence frequency is computed between two concept c_1 and c_2 and the contingency table is modified for $\{c_1, \neg c_1\}$ to $\{c_2, \neg c_2\}$. As proposed by Lim *et al.* [89], $\chi^2_{c1,c2}$ is normalized with $\left(\chi 2_{c1,c2} \middle/ \chi 2_{c1,d}\right)$ as a ratio between c_1 to c_2 and c_1 to domain of the concept c_1's dependency.

Table 4.5: *Semantic Concept Relations*

Relation type	***Linguistic Relation***
Equal	Synonyms
is-a	Hyponyms
has-a	Holonyms
part-of	Meronyms

v. Taxonomic Relation Extraction

In this phase of ontology learning, we have used a taxonomy induction approach that takes a set of domain concepts extracted in the previous step as input and generates a taxonomic relationship between domain concepts. In Table 4.5, semantic relation between concepts are listed with their linguistic relations. For taxonomic induction, hierarchical clustering is a

general family of clustering algorithms that build nested clusters by merging or splitting set of concepts successively. This hierarchy of clusters is represented as a tree to represent concept tree in which root of the tree is the unique cluster that gathers all concepts, and the leaves being the clusters with only one concept. Hierarchical clustering approaches generally fall into two categories named as, agglomerative (bottom up) and divisive (top down) and we adopted agglomerative clustering extended with preprocess (Line 2 Algorithm 4.16) and postprocess (Line 20 and 21 of Algorithm 4.16) to refine the taxonomic induction.

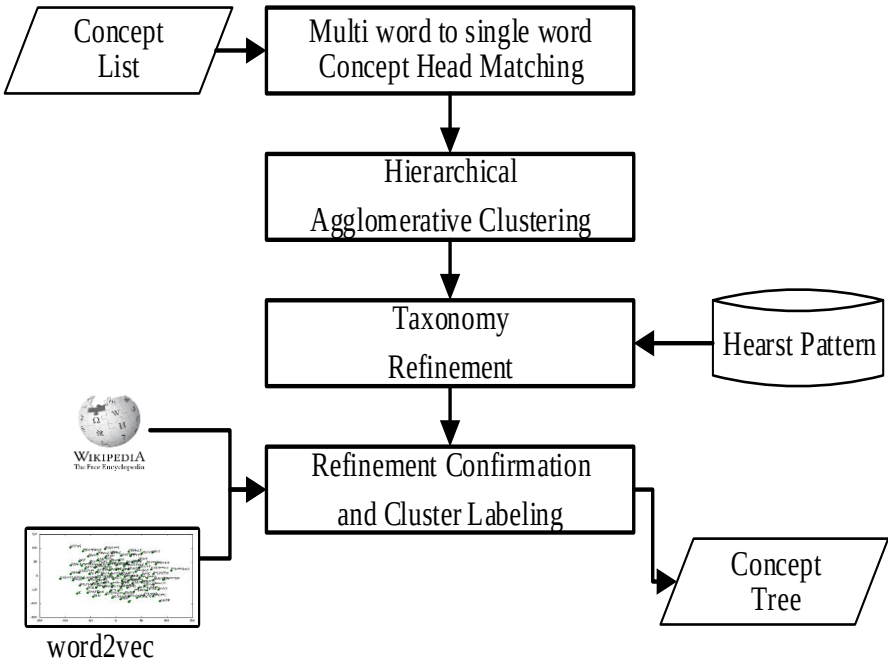

Figure 4.11: *Taxonomic Induction Workflow*

Conducted researches on ontology learning shown that the hierarchical agglomerative clustering (HAC) is an important and well-established unsupervised machine learning technique used for concept hierarchy extraction [117]. The adopted HAC starts from the partitioning of concepts into singleton clusters and merge step by step the pair of semantically closest clusters into a new node until there is only one final cluster left that contains the entire concept set. Thus, a hybrid of HAC and pattern with extended word2vec for taxonomic relation extraction refinement is proposed as shown in Algorithm 4.16. As pre-phase to HAC, head matching (Line 2 of Algorithm 4.16) is used to create taxonomic link between multi word and single word concepts. In agglomerative clustering, different linkage methods named

as *single*, *average*, and *complete* are used with different distance metrics such as *Euclidean*, *Manhattan*, and *Cosine*. In this work, merging a pair of clusters is done using the average linkage method (Line 6-15 of Algorithm 4.16).

Algorithm 4.16: *Taxonomic Relation Extraction Pseudocode*

```
Input: C:{c1, c2, c3,. . ., cn}//Concept list,
     W2VM: Word2vec Model and
     IARP: IS_A Relation Pattern
Output: Taxonomically Related Concepts
Begin:
1.   C ← ∅, Cindex=t, Cindex_Values=[ ], cluster_level=0,
     NumberofClusters=|C|
2.   C=HeadMatching(C)
3.   Assign each concept ci into its own cluster Ci and C=C U Ci
4.   cindex=computeCindex(cluster_level)
5.   Cindex_Values.add(cindex, cluster_level)
6.   Repeat:
7.        AverageSim ← ∅
8.        ClusterSimList = AvgClusterSim(ci, cj, cluster_level)
          // ci,cj in C computer average similarity
9.        Xci,cj= Max(ClusterSimList)  /* choose highly similar clusters*/
10.       Ck←merge(ci,cj) //Merge the similar clusters Ci and Cj
11.       cindex=computeCindex(cluster_level)
12.       Cindex_Values.add(cindex, cluster_level)
13.       cluster_level ++ // Increment cluster level
14.       C← C U Ck; C= C\Ci U Cj
15.  Until |C|== 1
16.  TopCindex=Cindex_value.top()
17.  Cuttinglevel= TopCindex.cluster_level()
18.  Cut the tree at cutting level  - Cuttinglevel
19.  //Taxonomy refinement
20.  CandidateRelations=Pattern_is_a_extraction(IARP)
21.  Confirmed_is_a = is_a_confirmation(CandidateRelations, W2VM)
End
```

In HAC, the clustering has to be examined and terminated before forming a single cluster, as the bigger cluster does not provide meaningful information and the overall taxonomic extraction workflow is depicted in Figure 4.12. The relevant HAC is decided with the clustering level (Line 16-17 of Algorithm 4.16) computed using C-index [118]. As cluster distance measurement metrics, word2vec based cosine similarity, chi-square, and PMI are used to measure (dis) similarity between pairs of clusters/concepts. The post-phase to HAC is

pattern based *is_a* relations extraction refinement that could be confirmed by word2vec (Line 20 and 21 of Algorithm 4.16).

Algorithm 4.17: *Cluster Labeling Pseudocode*

```
Input: CC: an internal cluster node {c1, c2, c3 . . . cn}
Variable: Vt: Label to Member Variation Threshold
   BV: Benchmark Vector // vector 1 to each dimension
   WD: Wikipedia Dump
Output: CL: Cluster Label
Begin:
1.  For mc in CC // for each element of a cluster node
2.    mcg = ExtractGlossfromWiki(mc, WD)  // wiki based
3.    isaflag=CheckIsAPattern(mcg)
      // check existence of is_a_pattern in the gloss
4.    If isaflag then // add the candidate to candidate gloss list
5.      CGL.add(mcg)
6.  End for
7.  If |CGL| > 0 // if there is a candidate gloss
8.    TopGloss = CGL.ExtractBestGloss() // identify the best gloss
      from the gloss candidate vector
9.    GT= TopGloss.ExtractGeneralConcept() // parent concept
10.   gt_vote=ComputeLabelVote (GT, CC) //using Algorithm 4.17
11.   If gt_vote > |CC|/2 and gt is not used as cluster label
12.     CL=GT
13.       else  Go to 15
14. else // if there is no valid wiki based gloss
15.   LT=LexicalBasedLabelExtraction(CC)
      /*multi-term based Label extraction – identify the head word
      for the terms in CC */
16.   lt_vote= ComputeLabelVote (LT, CC)//using Algorithm 4.17
17.   If lt_vote > |C|/2 and gt is not used as cluster label
18.     CL=LT
19.   Else
20.     CL = WikipediaOrientedClusterLabeling(CC)
        // Frequency is used as parameter to determine label
21.   Return CL
End
```

Mikolov *et al.* [26] demonstrated the success of word2vec analogy to answer unknown pattern value, i.e., if there is a pattern of the form *a is to b*, then one can associate *c is to d* where *d* is unknown, by finding the embedding vectors of va, vb, vc (all normalized to unit norm).

Following Mikolov *et al.* [26] approach, we adopt word2vec analogy to compute a new vector of d in the form:$y = vb - va + vc$ where, y is the continuous space representation of the word expected as the best answer for the analogy using Equation (7) [26].

In line with this, a very common method (and also one of the oldest) for relation extraction is the use of lexico-syntactic patterns, first described by Hearst [119]. These patterns compare sequences of words in the text against general patterns with naive approach towards relation extraction. In the last step of Algorithm 4.16, patterns based semantic relations are used as a concept hierarchy refinement. Thus, for pattern based relation refinement, common domain independent Hearst [119] patterns are used. However, to solve the drawback of pattern based approach, google semantics is used as a solving mechanism with the help of probabilistic counting of candidates using PMI based on Google search engine hit counts. As a result, the extractions have a form of tuple t= (c_i, r_{ij}, c_j), where c_i and c_j are strings meant to denote concepts, and r_{ij} is a string meant to denote "*is-a*" relationship between the two concepts. As seen in Equation (8) [26], similarity between two concepts can be confirmed using word2vec.

$$w^* = \text{argmax}_w \frac{x_w y}{\|x_w\|\|y\|} \tag{7}$$

Where:

- w* is concept, x_w with greatest similarity to y

Algorithm 4.18: *Compute Label Vote Pseudocode*

```
Input: GT: General term
   CC: Cluster Members = {c1, c2, c3 . . . cn}
Variable: Vt: Variation Threshold // Default is 0.5
Output: CL: Vote Count
Begin:
1. CLv=BuildVector(GT)// using word2vec model
2. For mc in cluster CC
3.   GT_MC_sim=ComputeSimilarity(CLv, BuildVector(mc))
       // vector based similarity is used
4.   If GT_MC_sim > Vt  then lt_vote = lt_vote+1
5. End for
6. Return lt_vote
End
```

Once the HAC is determined, we have to label each of the internal nodes using Algorithm 4.17. The algorithm accepts internal node and label it using members gloss definition (Line 1-12), members lexical matching (Line 15-18 of Algorithm 4.17), and member frequency (Line 20 of Algorithm 4.17) as labeling techniques. Algorithm 4.18 is used to compute the number of members that indorsed the candidate label (Line 5-6 of Algorithm 4.18).

$$\text{Similarity}_{(v1,v2)} = \cos(\theta) = \frac{\sum_{i=1}^{N} V_{1,i} V_{2,i}}{\sqrt{\sum_{i=1}^{N} V_{1,i}^2}\sqrt{\sum_{i=1}^{N} V_{2,i}^2}} \quad (8)$$

Where:

- *N* is the number of dimensions used for each word representation in word2vec model and
- *(v1, v2)* are vector of the two words used for semantic distance computation.

vi. Non-Taxonomic Relation Extraction

The two main issues in the extraction of non-taxonomic relations are relation discovery and relation labeling. The classical discovery process is based on the assumption that verbs indicate semantic relations. In this work, we proposed two algorithms for the extraction – Correlation based (Algorithm 4.19) and Analogy based (Algorithm 4.20). The correlation based approach, Algorithm 4.19, starts by identifying candidate verbs of a pair of concepts (Line 3 of Algorithm 4.19) and takes the one with a maximal dependency value (Line 4 of Algorithm 4.19). The verb becomes the non-taxonomic relation if its expectation value is above the minimal value (Line 6 of Algorithm 4.19) of being a relation label. Each occurrence of a relationship has the form of a tuple $\{<c_i, c_j> \rightarrow v \mid c_i, c_j \varepsilon C$ and v is a verb$\}$, where C is a set of domain concepts.

To validate concept pair to verb dependency above expectation (AE) measure is computed as defined in Equation (9) to test the association between concept pairs and candidate verb. If the co-occurrence of a concept pair *(c1, c2)* with a given verb *v* is more frequent than the individual concept's co-occurrence with *v*, then verb *v* is probably relates the concept pair, and thus *v* is considered as a candidate label for the concept pair. Moreover, for other types of semantic relations, such as meronym and holonym, an extended word2vec analogy based

neural technique is proposed as shown in Algorithm 4.20 and linguistic patterns are used as a pre-task for analogy based non-taxonomic relation extraction using word2vec analogy.

Analogy based non taxonomic relation extraction exploits the similarity of word representations in word2vec model that goes beyond simple syntactic regularities, using "*word-offset*" technique where simple algebraic operations are performed on the word vectors to find unknown words related with known words in vector representation, for example, *vector ("king") - vector ("man") + vector ("woman")* results in a vector which is closest to the vector representation of the word *Queen* [120].

Algorithm 4.19: *Correlation based Non-taxonomic Relation Extraction Pseudocode*

```
Input: CP: Concept pair list
   V: Relevant domain Verbs list and Ct: Correlation threshold
Variable:
   AEt: AE threshold
Output: Labeled non-taxonomically related pairs
Begin:
1. For each pair p in CP// result from concept pair extractor
2.   ComputedValues ← ∅
3.   ComputedValues = IdentifyCandiateVerbs(p)
     /*Identify the candidate verbs (in root form) relating concept
     pair p – using χ2/
4.   TopVerb= ComputedValues.Top() // get the top candidate
5.   AE= ComputeAE(p, TopVerb) // using equation 9
6.   If AE > AEt
7.   p.Relate(Label)
8. End for
End
```

$$AE(p,v) = \frac{\chi^2(p,v)}{\chi^2(c_i,v) * \chi^2(c_j,v)} \quad (9)$$

Where:

- *p* is concept pair (c_i, c_j)
- χ^2 chi-square compute concept pair to verb dependency
- AE is above expectation between p and verb v

Algorithm 4.20 uses pattern based relation extraction approach to identify a possible relationship of a given semantic relation, sem_relation, (Line 1 of Algorithm 4.20), applying

Hearst relation patterns. Then detects the best fit related pair of a concept and associate the domain and relation (Line 3-9 of Algorithm 4.20). CreateAnalogy is used to build a question that uses the relationship between *domain* and *range* as sample and demand what can be the association for c. In word embedding space, there is a consistent difference vector between male and female version of words. Similarly, in part-whole space, there are consistent features distinguishing between the part and the whole. As given in Algorithm 4.20, non-taxonomic relation requires Hearst's pattern for meronym and holonym relations as given input that used to formulate the analogy question.

Algorithm 4.20: *Word2vec Analogy based Relation Extraction Pseudocode*

```
Input: C: Set of Concepts
   RP: Set of Rules // (c.f. Hearst patterns of Non taxonomic Relation)
   RCP: Related Concept Pair, W2VM: Word2Vec Model
Output: Non-taxonomically related concept pairs
Begin:
1. RCP=PatternBasedRelationExtraction(RP, sem_relation)
   // candidate relations for semantic relation from a given corpus
2. For each concept c in C
3.   TSet=Concept2TermMapping(c) // concept to terms
4.   For each related pair rp in RCP
5.     Sim_rp_c = Similarity (TSet, rp) // vector based similarity
6.     Sim_rp_cList.add(Sim_rp_c)
7.   End for
8.   TopPair= Sim_rp_cList.Top() // take the top relation
9.   Domain, Range= ExtractDomainRange(TopPair)
10   Analogy=CreateAnalogy(Domain, Range, c)
11   RelatedConcept= W2VM.query(Analogy)
12   c.relate(RelatedConcept, TopPair.semanticRelation())
13 End for
14 Return C
End
```

4.3 Semantic Annotation Workflow Summary

As shown in Figure 4.13, a summary of the flow of tasks for performing semantic annotations of a given document is presented from text extraction up to the final annotation persistency. This workflow is divided into two main phases: (1) analysis phase and (2) annotation and storage phase. This semantic annotation workflow will be aborted due to the following

reasons, unknown document format, unsupported language documents, previously annotated documents, and unverified annotations as shown below.

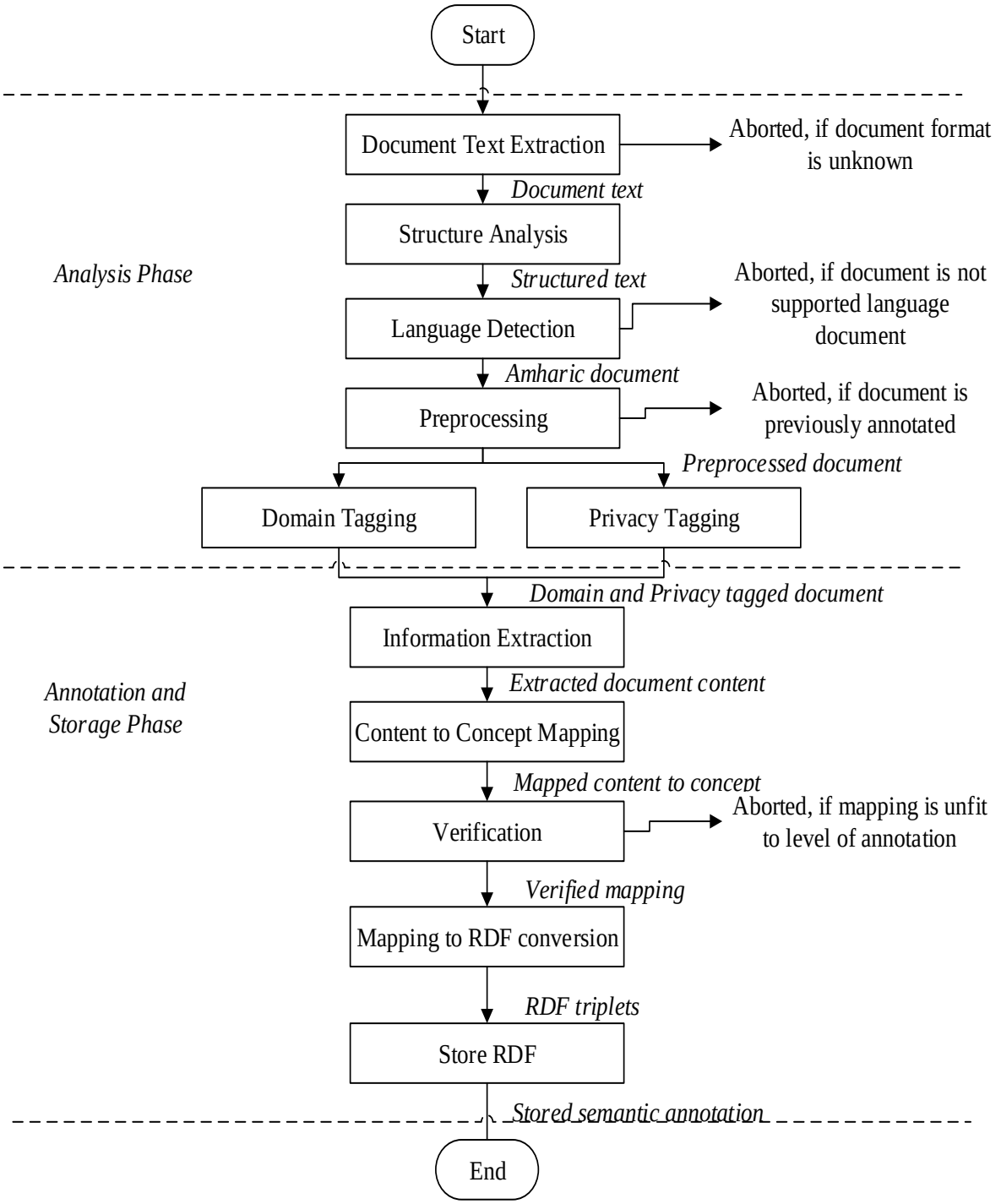

Figure 4.12: *Semantic Annotation Workflow Summary*

Chapter 5

Experiment

5.1 Introduction

Experimentation is conducted in order to justify the proposed generic document semantic annotation framework using Amharic natural language. Under this chapter, Section 5.2 presents experimental setups including dataset, tools and settings used for this experiment. Section 5.3 and 5.4 present semantic annotation and ontology learning experiments with their experimental result and evaluation. Afterwards, a discussion is presented in Section 5.5 by interpreting the result obtained from the experimentation.

5.2 Experimental Setup

5.2.1 Datasets

Amharic Wikipedia dump and POS-tagged Walta[22] news are used in different phases of document analysis, semantic annotation, and ontology learning. Although corpora used for machine learning tend to be bigger, we decided to rely on Walta news and Amharic Wikipedia dump, a middle-size corpus made of news articles and Wikipedia articles respectively. This choice is motivated by the fact that collecting huge sets of textual data is not always possible for all domains and languages.

Table 5.1: *Statistics of the Corpora*

Corpora	***# News/Articles***	***# Sentences***	***# Words***	***#Vocabulary***
WALTA Tourism domain News Corpus	46 News	423	9244	3829
All WALTA News	1103	10155	222355	32413
Wikipedia Dump Corpus	13976 Articles	-	>3 mil	>50,000

The POS-tagged Walta news dataset is containing 1103 news articles categorized manually by domain experts into 7 domains, while the dump[23] of Amharic Wikipedia contains

[22] http://waltainfo.com

[23] https://dumps.wikimedia.org/amwiki/

Wikipedia articles with a good coverage of all major Amharic relevant topics in more than thirteen thousand articles as it can be seen from Figure 5.1 of the Wikipedia preprocessing screenshot and statistics of both Walta news and Wikipedia dump are shown in Table 5.1. The dumped raw Wikipedia XML corpus was preprocessed with WikiExtractor[24], a Python script that extracts and cleans text from Wikipedia XML Dump and Figure 5.1 shows the preprocessing information of the Amharic Wikipedia Dump.

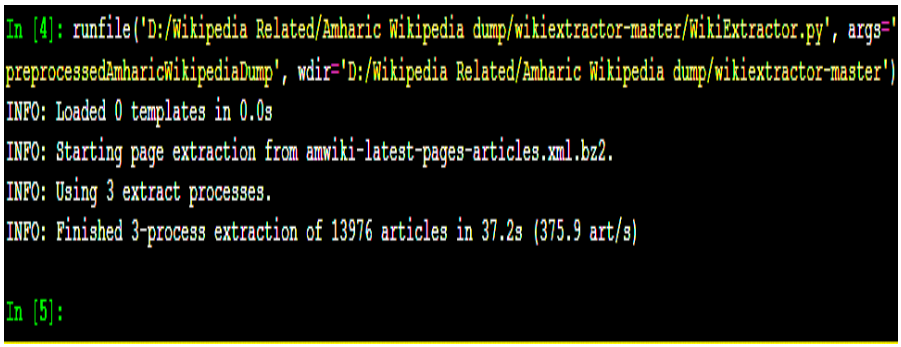

Figure 5.1: *Amharic Wikipedia Dump Preprocessing Result*

5.2.2 Packages and Tools

In order to experiment the proposed solution, different appropriate packages and tools are selected and employed. Moreover, different custom implementations of document analysis, ontology learning and semantic annotation of algorithms proposed in the previous chapter are also experimented and tested using the Python programming language. Table 5.2 shows a list of packages and tools used in our experiments with their versions and descriptions.

Table 5.2: *Description of Packages and Tools*

Packages/Tools	Version	Description
Anaconda (Spyder)	3.1.2	Spyder stands for Scientific PYthon Development EnviRonment included in Anaconda distribution.
Django	1.10.5	Django is a web application framework written in python used for creating user interface.
gensim[25]	0.13.3	Python package that implements word2vec for word embeddings and document to vector conversion.

[24] https://github.com/attardi/wikiextractor/wiki

[25] https://radimrehurek.com/gensim/

HornMorpho	2.5	Morphology analyzer and generator working for three local languages, including Amharic.
Keras	1.0.6	A Neural Network library written in Python, which is designed to be minimalistic and straight forward yet extensive that built on top of Theano and Tensorflow.
Langid	1.1.6	Language identification model that was pretrained for 97 languages, including Amharic.
Mysql	5.7	Mysql, popular and lightweight database software used for storage of annotated documents metadata, and "*pymysql*" python package was used to create connection between python and mysql.
NLTK	3.2.2	Natural language processing toolkit used for n-gram computation for multiword extraction.
Owlready	0.3.0	Owlready is an ontology package used for domain ontology manipulation.
PDFMiner3k[26]	1.3.1	Python package used to mine and extract texts and layouts from pdf documents.
TextBlob	0.12.0	TextBlob is a Python library for processing textual data and provides a simple API for natural language processing (NLP) tasks such as sentiment analysis, classification, and translation.
Wikipedia API	1.4.0	To access and parse data from Wikipedia

5.2.3 User Interface

User interface is highly desired by document owners and knowledge manager to be successful in contributing to semantic web, through semantic annotation. Interactive and easy to use user interface is required. The design of such interface should consider how the target users actually generating semantic data of their document easily from anywhere, and user interface shown in Figure 5.2 is a web based interface that allows the target user to upload and annotate his/her document easily from a web browser. In order to submit document for annotation, either

[26] http://www.unixuser.org/~euske/python/pdfminer/

uploading document or pasting text (Figure 5.3) can be used. In both cases, the document owner has to be registered and expected to set the level of annotation with privacy preference. If the user skipped over the annotation level, the system is enforced to use the default settings (i.e., moderate level of annotation).

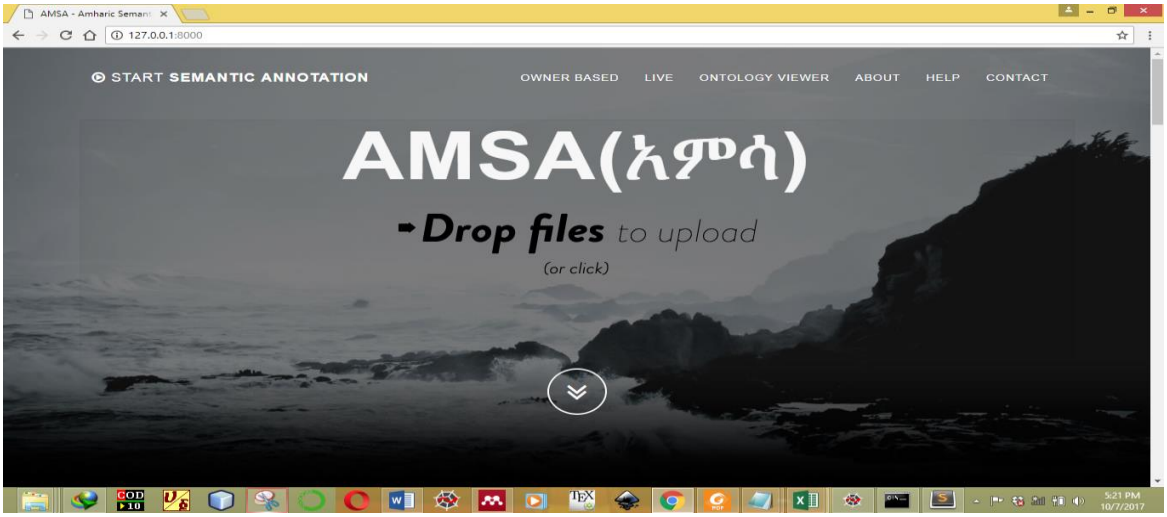

Figure 5.2: *Homepage User Interface for Amharic Semantic Annotator*

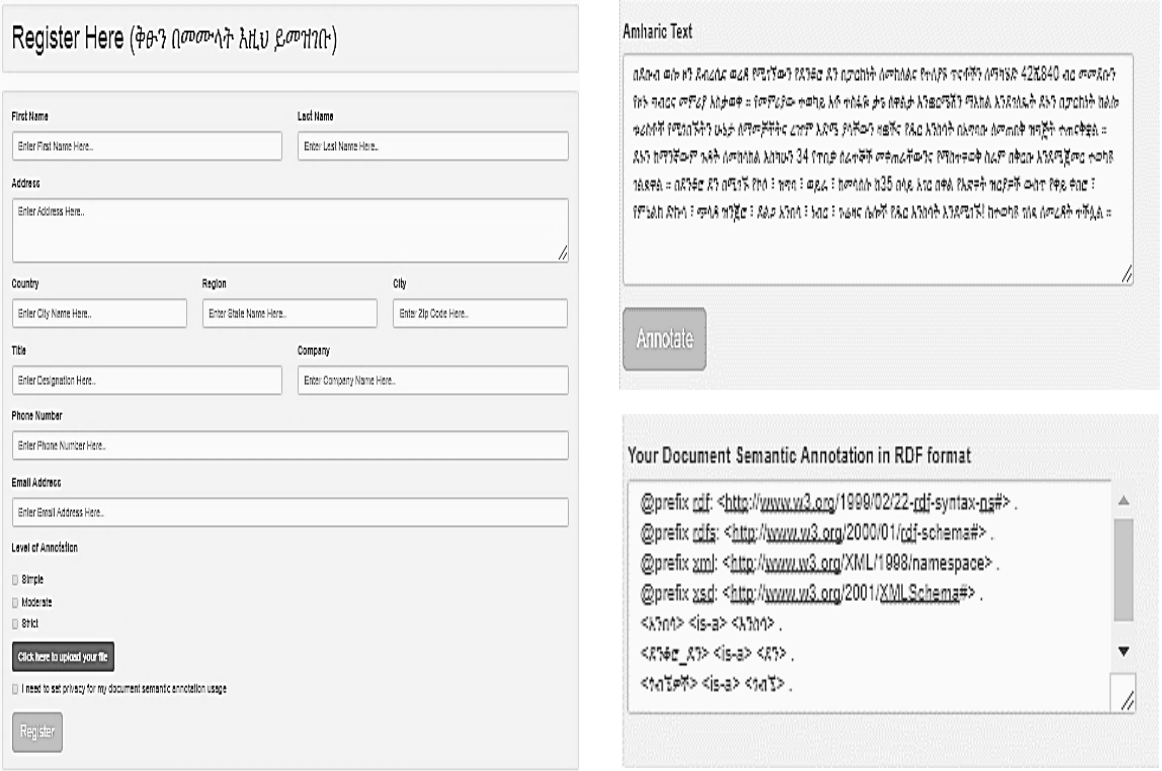

Figure 5.3: *Sample User Interfaces*

As shown in Figure 5.3, the user interface allows the document owner to upload his/her document to be annotated with other personal (e.g., name, email) and preference (i.e., level of annotation and privacy tagging option) information. Once the document is uploaded, the analyzer component of the framework analyzes it and sends to semantic annotator, and finally the result of annotation is displayed in RDF format to the owner of the document as shown in Figure 5.3.

5.2.4 Experimental Setting

In experimental tasks, execution time is affected by several factors, including hardware features, single thread or sequential execution of tasks, and dataset size. In this work, we have implemented several experiments. All experiments were executed on Toshiba Laptop with Intel core i3 and 4GB RAM running Windows 8 Operating system with a network connection speed of 54.0 Mbps. Thus, execution time will be improved using high performance hardware and multithreaded execution for the learning and annotation tasks.

5.3 Experiments

To determine the success of the proposed solution, four general experiments named as document analysis, word embeddings, ontology learning and semantic annotation are conducted and an overview of each experimentation is shown in Table 5.3, detailed procedure is presented under each subsection.

Table 5.3: *Overview of the Experiments*

Experiments	***Description***
Document Analysis	Experimenting document structure analysis, language detection, document preprocessing with domain and privacy tagging using custom code and python document analysis python packages
Word Embeddings	Train and test word2vec model using gensim python package
Ontology Learning	Experimenting term extraction, concept to domain and concept to concept mapping, taxonomic and non-taxonomic relation extraction to model ontology
Semantic Annotation	Experimenting information extraction, content to concept mapping, verification and annotation maintenance

5.3.1 Document Analysis

Under this experimentation, first document format has to be detected using its extension and appropriate python packages were used for the textual content extraction. Hence, PDFMiner, lxml.html, docx, and xmltodict python packages are used to convert document in pdf, html, word, and xml format into textual format respectively. The next step is identification of the main structural elements of the document, in particular, sections such as title, abstract, introduction, paragraphs, sentences, conclusion, references, and other document metadata in order to fill the metadata/structural template. The extraction and template filling were performed with the help of different parameters, including, newline, indentation, text position in a document, number of words and subtitles as decision criteria.

After the structure of the document is analyzed, language detection is experimented using LangID, a pre-trained language identification model of 97 languages. The detection model is loaded to categorize a given document into one of 97 languages provided in Annex B using byte n-gram and naïve bayes classification approach using its python script. Once the document language is detected, natural language preprocessing tasks, including tokenization, and sentence splitting are performed with redundancy checking and character mapping for the normalization of the input document with the help of records found in normalization and delimiters record repository.

In order to check redundancy, first the document is converted into vector using gensim *doc2vec* python module and annotated metadata repository is queried for the converted vector to check either the document is previously annotated or not. If the document is new for annotation (i.e., its vector is not found within the annotated documents metadata repository) its vector representation is tagged to the document and proceeds to the annotation process, else redundancy flag is changed to true, which indicates that a given document is not new for annotation, and the process of document semantic annotation is aborted. For document detected as Amharic document and new for annotation, domain categorization of Amharic document is experimented using Walta news labeled into seven domains named as: *Tourism*, *Business*, *Education*, *Entertainment*, *Politics*, *Social*, and *Sport* with statistical information presented in Table 5.4. To train the categorical model the labeled news dataset was prepared in a JSON file format as provided in Annex C and "*textblob*" python package is used for

training and testing of the categorizer. The trained model is saved as *pickle* (i.e., a file format used to save model) format for the sake of persistency and loaded later to categorize the domain of unknown documents.

Table 5.4: *Statistics of News Articles used for Domain Categorization*

S.NO.	***Domains***	***Number of News***
1.	Tourism	20
2.	Business	16
3.	Education	20
4.	Entertainment	7
5.	Politics	19
6.	Social	19
7.	Sport	8

Next, privacy tagging is experimented for knowledge based privacy tagging using gazetteers containing a list of keywords (from knowledge base repository) mainly occurring in private documents. As a sub-process of privacy tagging, for document labeled as private, either by the owner of the document or existence of privacy triggering keywords, privacy key is generated with the help of "*Crypto*" python package that generates RSA based key with given number of bits (e.g., 1024 bits). The generated key contains both private and public key and "*key.publickey*" function was used to extract the public key from the generated key and tag the document.

5.3.2 Word Embeddings

This section presents experimentation of word embeddings (word2vec) using gensim python package for word2vec modeling (sample code is provided in Annex I). This package is used to build word embeddings models from Wikipedia and Walta news dataset with word embeddings training settings shown in Table 5.5. As it can be seen from Table 5.5, for training Walta corpus, we set the word2vec "*min count*" variable to one (which means that all tokens will be considered) whereas for the Wikipedia dataset it was set to three (which means only words occurring at least three times are considered). After training word2vec, visualization of the 63-dimensional vectors of Tourism domain news illustrated in Figure 5.4 in two

dimensional embeddings space using t-SNE[27]. In Figure 5.4, the neighboring dots show words close to each other semantically. This trained model is later used for different tasks under ontology learning and semantic annotation due to its syntactic and semantic information capturing ability. As an illustration, list of nearest neighbors in the word embedding space for three words are listed in Table 5.6.

Table 5.5: *Word embeddings Training Settings*

Training Corpus	***Model***	***Dim[28]***	***Min count[29]***	***Windows Size***	***Size of Vocabulary***	***Skip-gram/ CBOW***
Wikipedia dump	Word2Vec	190	3	5	35756	Skip-gram
WATA News	Word2Vec	63	1	3	3826	Skip-gram

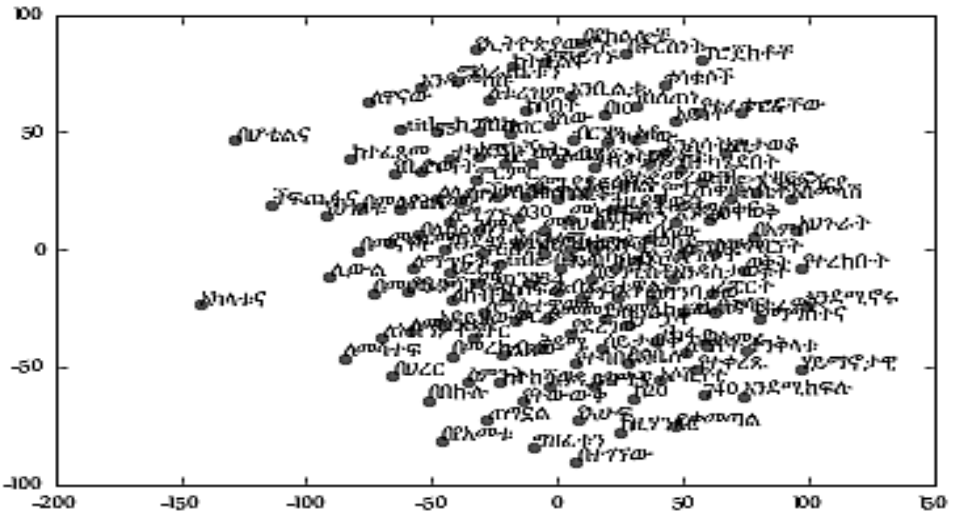

Figure 5.4: *Tourism Domain Word2Vec Model Visualization using t-SNE*

Table 5.6: *Nearest Neighbors in the Word Embedding Space*

Words	ተራራ	ደሴት	ቻይና
Nearest Words	ምድር	ወደብ	ሃገር
	ደን	አጠገብ	ሶርያ
	እንስሳት	አምባ	ንግሥና

[27] T-SNE, t-distributed stochastic neighbor embedding, is used to visualize multi-dimensional vectors in 2-D space.

[28] A good heuristic that frequently used for dimension value is approximate to square root of the length of the vocabulary, usually more dimension is better, but not always.

[29] Used to reduce noise in the semantic space.

5.3.3 Ontology Learning

This section presents experimentation of ontology learning phases from concept extraction up to semantic relation extraction and graph generation (see Annex I for sample code). In order to extract candidate domain concept sets, concept extraction phase of ontology learning was experimented using TF-IDF approach, followed by MWE, synonymity computation, and Wikipedia usage. Table 5.7 and 5.8 show sample results of concept extraction phase. As end process, stemmer[30] is used to group candidate concepts sharing the same root and having different formations and derivations.

Table 5.7: *Sample Extracted Single Term Concepts with help of Wikipedia Infobox*

S.No	*Extracted Terms*	*Transliterated Terms*	*English Meaning*
1.	አጥቢ	ät'əbi	Mammalian
2.	ተራራ	Tärara	Mountain
3.	ገዳም	gädam	Monastery
4.	ጥርስ	t'ərəs	Teeth
5.	ዝርያ	zərəya	Species/Family

Table 5.8: *Sample Extracted Multi Word Concepts*

S.No.	*Multi Word Concepts*	*Transliterated Concepts*	*English Meaning*
1.	ቅርሳቅርስ አስመላሽ	qərəsaqərəs äsəmälaš	Heirlooms Returner
2.	አብያተ ክርስቲያናት	äbəyatä kərəsətiyanat	Churches
3.	ጭላዳ ዝንጀሮ	č'əlada zənəğäro	Bleeding-heart Monkey
4.	ቤንሻንጉል ጉሙዝ	benəšanəgul gumuz	Benishangul Gumuz (Region in Ethiopia)
5.	እንፎርሜሽን ማእከል	ʾənəforəmešən maʾəkäl	Information Center

After these processes executed, the performance of the learner at concept extraction phase was evaluated for extracted concepts provided in Annex G as a sample and see Section 5.4.1 for concept extraction precision values. After concept extraction, the ontology learner experimentation was proceeded into taxonomic relationship induction between concepts. The

[30]Amharic stemmer of Tessema Mindaye, Hassen Redwan and Solomon Atnafu is adopted for stemmer oriented experimentation of this work.

taxonomy induction starts with head matching for multi word based taxonomic grouping as a prephase to HAC. The sample result of the tourism domain concept taxonomy is illustrated in Figure 5.5. We used custom code for HAC concept taxonomy generation in which list of concepts, similarity matrix were used with c_index computer as a parameter. After, executing HAC for top 50 tourism domain concepts, 49 taxonomic relationships were created as shown for sample in Figure 5.5 and its detail is provided in Annex H. In this experiment, sample Hearst pattern shown in Table 5.9 is used as a taxonomic relation refinement, and confirmed using word2vec.

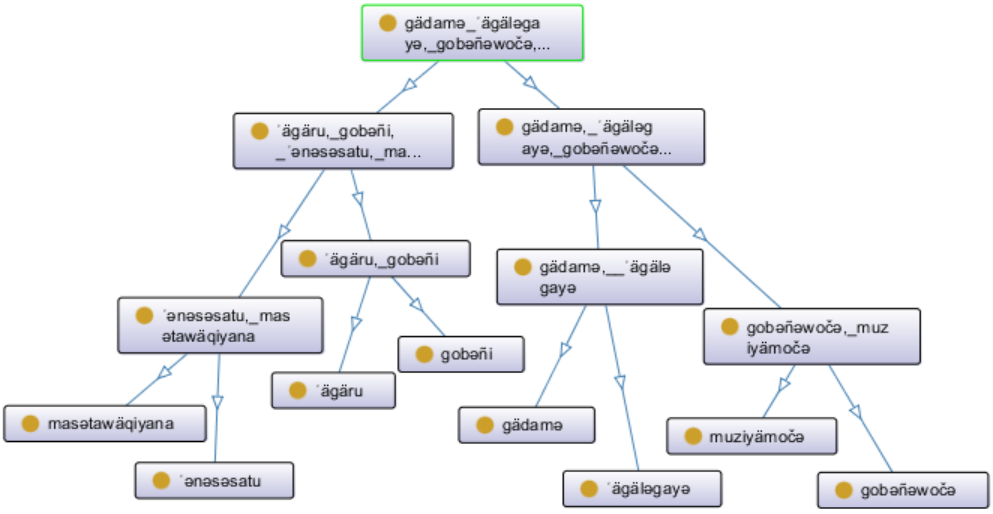

Figure 5.5: *Sample Tourism Domain Ontology*

Table 5.9: *Sample List of Patterns used for Taxonomic Relation Refinement*

S.NO.	***Patterns***	***Example***
1.	C_x የሚባል C_y	ለይንስ ጌት ***የሚባል*** ኩባንያ *(ለይንስ ጌት, ኩባንያ)*
2.	C_x C_y ነዉ | ዘርፍ ነዉ| ናቸዉ	ተአጥሮ ሳይንስ **የሳይንስ *ዘርፍ ነዉ*** *(የተአጥሮ ሳይንስ, ሳይንስ)*
3.	C_x የC_y (ዘር|አይነት) ነዉ	እንጀራ የምግብ *አይነት ነዉ (እንጀራ, ምግብ)*
4.	C_x እንደ አንድ C_y	አዲስ አበባ ዩኒቨርሲቲ ***እንደ እንድ*** ዩኒቨርሲቲ *(አዲሰ አበባ ዩኒቨርሲቲ, ዩኒቨርሲቲ)*
5.	(እንደ (C_x ፣)* (እና |ወይም) C_y) ያሉ (C_z)	***እንደ*** ሩጫ ***ያሉ*** የስፖርተ ዘርፎች *(ሩጫ, ሰፖርት)*
6.	(C ፣)* (እና | ወይም) ከሌሎች | ከተመሳሳይ C_y	ደቡብ ክልል ***ከሌሎች***ክልሎች ጋር (*ደቡብ ክልል, ክልሎች*)

Next to taxonomic induction, non-taxonomically related concepts were identified as shown for sample in Table 5.10. For non-taxonomic relation extraction, two approaches (correlation based and word2vec analogy based) were experimented to extract set of triples ($concept_i$, relation, $concept_j$) relating domain concepts. In correlation based non taxonomic relation extraction, concept pair list, verb list, and dependency threshold were given to the function as an argument and the function computes pair to verb dependency to relate the pair into the domain verb with a higher correlation dependency value to the pair. Thus, concept pair is related with top relating verb from verb list, if its dependency value exceeds a given correlation threshold value.

Table 5.10: *Sample Result for Correlation based Non-Taxonomic Extraction*

Relation Label		***(Domain, Range)***	
Label in Amharic	*Transliterated Label*	*Domain-Range*	*Transliterated Domain-Range*
ነበኙ	Gobäñu	(ነብኚ, ደን)	(gobəñi, dänə)
መነበኘት	mägobäñätə	(ቅርስ, ነብኚ)	(qərəsə, gobəñi)
ገልጸዋል	gäləs'äwalə	(አገልጋይ, ገዳም)	(ʾägäləgayə, gädamə)

Moreover, for word2vec analogy based relation extraction, sample Hearst patterns shown in Table 5.11 and 5.12 are used for analogy formulation. Thus, for word2vec analogy based semantic relation extraction, word2vec model is loaded using "*load*" function of the gensim package. After the model is loaded, information was extracted from "*pattern_based_relation*" argument that has domain, range and semantic relation type. After extracting information, the relation analogy is created and word2vec model is queried to return a new concept that related to the given concept, in like manner a given domain is related to the range.

Table 5.11: *Sample List of Patterns for Meronyms (part-of) Relation*

Patterns	***Example***	***Relationship***
የC_x C_y	የመኪና ጎማ)	*part-of* (ጎማ, መኪና)
C_x C_yን ያካትታል	ቤት መኝታ ብትን ያካትታል	*part-of* (መኝታ ቤት, ቤት)
C_x የተሰራዉ ከ(C_y ፣)* እና C_z ነዉ	ጠረጴዛዉ የተሰራዉ ከእንጨት እና ብረት ነዉ	*part-of* (እንጨት, ጠረጴዛ)
(C_x ፣)* እና C_y C_z ስር ይገኛል | ይገኛሉ	የካ ክፍለ ከተማ በ አዲስ አበባ ስር ይገኛል	*part-of* (የካ ክፍለ ከተማ, አዲስ አበባ)
C_y የC_x አካል ነዉ	እጅ የሰዉነት አካል ነዉ	*part-of* (እጅ,ሰዉነት)

Table 5.12: *Sample List of Patterns for Holonym (has-a) Relation*

Patterns	*Examples*	*Relationship*
በC_x ሥር ያሚካተቱ | የሚገኙ	ተፈጥሮ ሳይንስ ስር የሚካተቱት ኮምፒዩተር ሳይንስ እና	*has-a*(ተፈጥሮ ሳይንስ, ኮምፒዩተር ሳይንስ)
C_x (C_y ፣)* (እና | ወይም) C_z አሉት	ዩኒቨርሲቲ ቤተመፃህፍት እና የመማርያ ክፍሎች አሉት	*has-a*(ዩኒቨርሲቲ, ቤተመፃህፍት)

5.3.4 Semantic Annotation

This section presents the semantic annotation process that starts with information extraction and ends with verification (see Annex I for sample code). NER is modeled for term level extraction from web document. The NER modeling dataset is derived from POS tagged Amharic Walta news and manually tagged with the help of linguistic experts. The NE tagged dataset is prepared in IOB2[31] data encoding format as provided in Annex F into three sets (training, validation, and testing set) with vocabulary size of 981 words. The dataset was labeled with tag sets listed in Table 4.1 (such as PER, LOC or ORG) including an O tag for words not classified under Table 4.1 named entity tag sets. In order to model NER, we used Keras deep learning library's for sequential modeling that used indexed word lists provided as a sample in Annex D. This model has three layers named as embedding input layer, LSTM-dense hidden layers, and activation output layer. Training of the model is performed with the help of "*fit*" function of the Keras deep learning library.

Sentence level tagging experimentation starts with feature vector computation of both abstract and conclusion section to choose the winners of the two (abstract and conclusions) in terms of relevancy using word2vec and doc2vec vectorization. Next, tagging process goes into sentence selection in the same fashion using the feature vector computation of each sentence and three terms named as subject, object and predicate are extracted from the representative sentence with the help of "*HornMorpho*" python package, Amharic morphological analyzer and positions of terms in a sentence.

In order to tag document paragraphs with their vector representation, doc2vec gensim module that computes a feature vector of a given paragraph is used with word2vec. In doc2vec

[31] IOB2, Inside-Outside-Beginning2, a common tagging format for tagging tokens in a chunking task of computational linguistics including NER.

implementation, there is a need to specify that how many numbers of words or sentences convey a semantic meaning. Thus, labels or tags to a paragraph are specified depending on the level of semantic meaning set to doc2vec and each paragraph tagged with its feature vector as (ParagraphID, *has_vector*, ParagraphVector) triplet and paragraph's *is_about* is tagged by querying the word2vec model for most similar concept using the *has_vector* of each paragraph.

To experiment content to concept mapping of the extracted document contents, Owlready python package is used to query the learned ontology and link the content with ontological concepts. The mapper searches ontology in order to lookup ontology for a given document content and link weight is computed based on the link type created between content and concept. Once each content has mapped into their related concept the mapper pass the mapping into an annotation verifier.

Verification is experimented using three parameters named as information weight, content to concept link weight, and level of annotation with the aim of approving the result of the semantic annotation in terms of user preference. The verifier computes information weight of contents and concepts using TFIDF and penalize the mapping with link type weight. We sum up each mapping weight and set the score of annotation as the normalized average value of the total mapping weight and the verification decision has made based on the level of annotation and computed score of annotation. If the verifier set the flag to true the mappings sent to the RDF serializer for persistency, otherwise the annotation is aborted due to unverified mappings. For serialization, RDFLIB, python API for RDF that contains parsing and serializing for rdf/xml, ntriples, turtle and other formats is used with its graph interface that has support for SPARQL queries and update statements. The rdflib based graph interface is created and each triplet is added into the graph using "*add*" function. After the graph is created for all mappings the "*serialize*" function accepts file destination and store semantic annotation of the annotated document in turtle format.

5.4 Evaluation

5.4.1 Ontology Learning Evaluation

The performance of the proposed approach for the ontology learning is dependent on the amount of data used in the learning process. Figure 5.6 shows the timing using varying sizes

of words in comparison to the total dataset used, i.e., 3754 (100%). The time increases linearly with size.

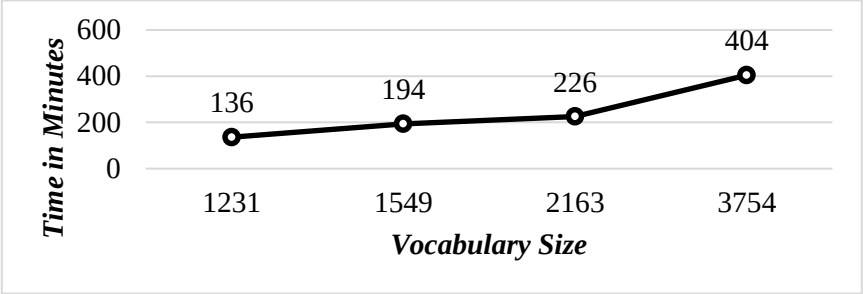

Figure 5.6: *Ontology Learner Execution Time Series*

The two commonly used ontology evaluation techniques are automatic evaluation using gold standard and manual evaluation using experts. Due to the absence of modeled knowledge for Amharic as a gold standard, expert based evaluation method is used for our ontology learner evaluation. Thus, to evaluate the quality of generated ontology, we let human experts, with a background in computational linguistics, to judge how far the extracted information is correct (i.e., the precision is measured) for the major phases of the ontology learner.

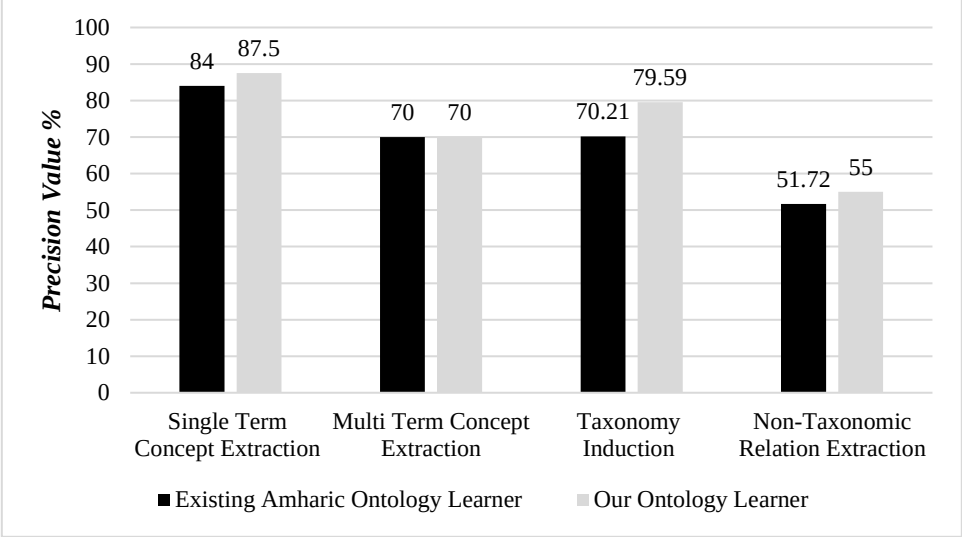

Figure 5.7: *Comparison of Existing and Our Ontology Learner*

This method of evaluation has a drawback, since the extracted information is not compared with the information found in the corpus, but with the knowledge of the human expert. This is not so problematic for measuring the precision of the learning algorithm, but it makes a reliable measurement of the recall nearly impossible. The result of the learner (provided as a

sample in Annex G and H) was provided to three linguistic experts and they made a judgment on the result at each level of ontology learning, i.e., concept extraction up to non-taxonomic relation extraction.

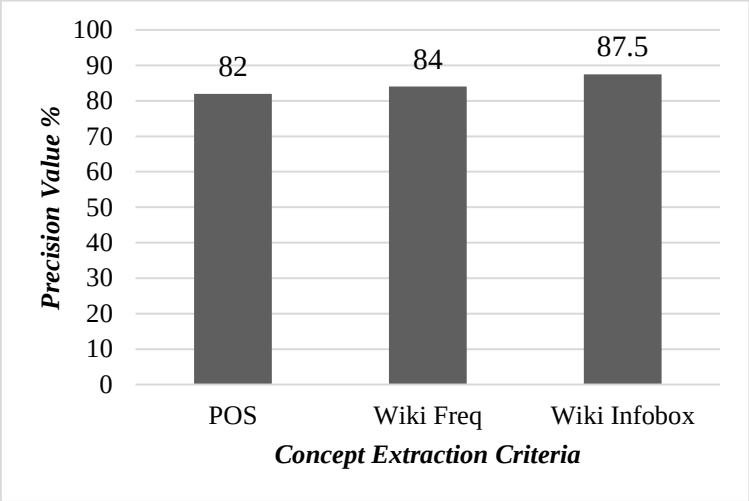

Figure 5.8: *Term Extraction Evaluation Result*

We applied majority voting approach, i.e., agreement between at least two experts in determining precision. The use of recall to measure relevance in this context is nearly impossible. The result of this work is compared against existing approach of Berhanu Mengiste [121] as shown in Figure 5.7 and our approach out performs it in all phases of the learning process using the same dataset under tourism domain. We have also evaluated the impact of NLP process, Wikipedia frequency and Infobox in the term extraction process. We found that the use of Infobox improves the result the greater extent as shown in Figure 5.8.

5.4.2 Semantic Annotation Evaluation

Semantic annotator is evaluated in terms of recall (i.e., to measure how much relevant document contents are extracted and mapped into ontological concepts), precision (i.e., to measure how much the extracted and mapped content is correct) and f-measure (i.e., combination of precision and recall). Expert based manually annotated news dataset is used as a "gold standard" to see how well the system performs with respect to the gold standard.

For the evaluation, from 46 news articles in tourism domain 23 news were selected using systematic sampling technique (i.e., sorting news with their number of words, and every news at odd rank index were selected as a sample). The sample news dataset were manually

annotated by three linguistic experts (i.e., convenient for inter annotator agreement). This manually annotated document was used as a gold standard for the proposed semantic annotator evaluation and Figure 5.9 shows the evaluation results of our semantic annotator.

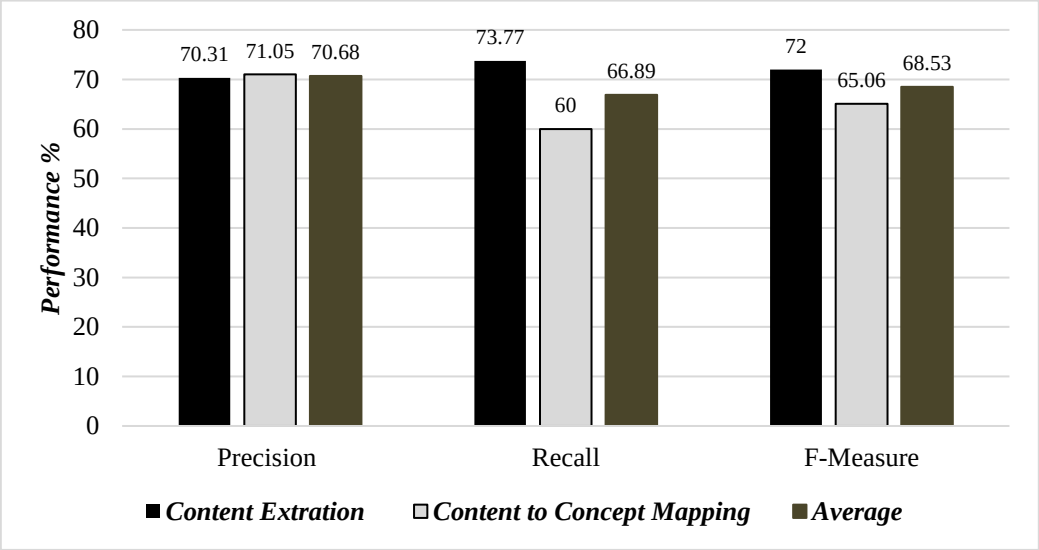

Figure 5.9: *Semantic Annotation Performance Evaluation*

5.5 Discussion

As shown in Figure 4.1, our proposed framework has several components and the performance of each component has an impact on the overall performance of the semantic annotator. In addition to this, to solve natural language processing and semantic understanding related problems, it is in need to use powerful models such as neural networks. If a model is not powerful model, then there is absolutely a challenge to succeed in a real world scenario. Hence, (deep) neural network is a powerful modeling technique adopted and experimented for different phases of semantic annotator and ontology learner with other document semantics understanding techniques.

Accordingly, in the proposed and experimented solution, document analysis and neural network techniques are used to minimize cost of annotation that comes due to unanalyzed document and maximally supervised language specific features respectively. In this work, though the experimentation is conducted for Amharic web document, the solution is proposed using generic methods working for other languages and domains with minimal modifications.

Furthermore, in neural network, one major requirement that all RNN models for NLP share is lemmatization especially for morphologically rich languages like Amharic. However, due to absence of publicly available lemmatization for Amharic, the effect of lemmatization on the overall performance of the semantic annotator is not experimented in the case of this research.

In ontology learning, several criteria have been used and each criterion has its own advantages and disadvantages in terms of processing time, genericity, and scalability. As its result is presented in Figure 5.7, usage of information from external corpora, Wikipedia in our proposed solution presented an improvement for ontology learning and outperforms the existing Amharic ontology learner presented in [121] following WaC principle.

In semantic annotation, the experiment and its evaluation shows promising solution; however, it is not sufficient for content to concept mapping evaluation, because the distinction between right and wrong is less obvious for content to concept mappings. For instance, mapping a "*person*" into "*location*" is clearly wrong, but mapping a "*research-assistant*" as a "*lecturer*" is not so wrong. Furthermore, size of the modeled ontology and accuracy of the extracted information are also other performance regulators of the content to concept mapper. Thus, information extraction and ontology content have to be improved for better performance result of semantic annotator.

In our modeling (LSTM for NER, word2vec for word embedding, naïve Bayes for domain categorization and others), word is considered as the fundamental and smallest units of language modeling. However, recent research directions for LSTM and word2vec enhancement have shown the necessity of considering smallest units beyond word level for better modeling of languages, particularly morphologically rich languages like Amharic. This enhancement improve the current word aware modeling (works well for morphologically simple languages) into morphology aware modeling (works better for morphologically rich languages). Thus, even if the adopted methods and techniques we used for both semantic annotation and ontology learning shown promising results, in order to improve and enhance our modeling in a way that works well for morphological rich languages like Amharic, morphemes and graphemes beyond word level have to be considered for information extraction, word embedding, and other processes considering word as smallest unit.

Chapter 6

Conclusions and Future Works

This chapter presents conclusions that discuss the activities done in this research work with how the problems are addressed and achieved the intended objectives. Moreover, future works that exceed the scope and findings of this study from different perspectives are also discussed under Section 6.2.

6.1 Conclusions

The aim of this research was to propose a semantic annotation framework for unstructured web documents using a range of complementary methods. In this work, in order to understand the foundational theories, approaches and techniques behind semantic annotation and ontology learning we have reviewed different literatures and presented their linkage into our research work. In line with this, different related works have also been addressed with their drawbacks. In the interest of the largely increasing documents over the Web and identified drawbacks in existing related work, (generic) semantic annotation framework is proposed for unstructured Web documents.

This framework considered different requirements and drawbacks of existing semantic annotation research efforts in its preprocessing, inprocessing, and postprocessing phases using generic methods that have feature of adaptability into other problem domains. To verify the proposed solution, four experiments were conducted using python programming language and predefined ML packages (like: gensim, keras, and word2vec). For this experimentation, we have collected textual corpora from Walta news agency and Wikipedia. The experimental evaluation exhibits 70.68% of precision, 66.89% of recall and 68.53% of f-measure for semantic annotation and the results show that the adopted and extended language independent techniques we used for both annotation and ontology learning are promising for understanding document semantics and learning domain knowledge with no (minimal) language specific features. Our experiments were carried out with limited size of corpus[32] (i.e., news articles from Walta news agency and Amharic Wikipedia dump) in comparison to corpus used by

[32] In corpus, number of documents (more documents are better) and similarity of documents within a single domain (more similar is better) is a factor determining quality of the corpus.

other researches adopted neural network and probabilistic models; however, our experimentation shows promising result for both semantic annotation and ontology learning. The current result of both semantic annotation and ontology learning would have had state of the art result if the dataset was large and consistent. In the context of this study, linguistically motivated methods, like pattern usage provide slightly better performance than the language independent approaches for non-taxonomic relation extraction of the ontology learning; however, this improvement comes at the cost of speed and challenge of adaptation into new languages and domains.

Extraction of semantic information and domain knowledge from unstructured web document is highly required for existing web documents understandability while ensuring the goal of semantic web and this study has made the following major contributions related to semantic annotation, ontology learning and NLP as bridging solution between existing web and future semantic web.

- Generic framework for semantic annotation with ontology learning integration.
- Neural networks and deep learning adaptation for ontology learning and information extraction of semantic annotation.
- Term, sentence and paragraph level content extraction and tagging of a given document to be annotated semantically.
- Structure-aware information extraction for semantic annotation.
- Domain and language detection as preprocessing tasks for semantic annotation.
- Adding a level of trust into semantic annotation of documents.
- Improved word2vec model usage for different tasks under semantic annotation and ontology learning.
- Inclusion of semantic annotation postprocessing (verifier), to score and verify the semantic annotation.
- Consideration of semantic annotation requirements discussed under Chapter 2 and Chapter 3.

6.2 Future Works

On the basis of the findings presented in Chapter 5, the following potential extensions that will enhance the performance, scope and outcome of the proposed solution are identified as future works.

- As the current experimentation is limited to Amharic document semantic annotation, the proposed framework adaptability for different languages (such as English, French and Afan Oromo) and domains shall be experimented and compared with the result of this research work.
- Framework extendibility from a textual document into multimedia document semantic annotation is also part of the future work by porting multimedia content analyzer into document analyzer comprehensive components of the proposed framework.
- The proposed solution focuses only on the textual content extraction in structure-aware manner and extended extraction methods that can handle multi-modal contents, such as images and graphs will be part of the future work.
- Extending existing ontology self-learning mechanism that depends on the context of new entities only from a given document into the task populating ontology with new entities using "*slot-filling"* techniques with the help of large textual corpus as a source of knowledge.
- Many word embeddings based NLP models achieved high performance using only words as inputs. However, the trained word vectors for rare words are usually poorly represented, since they do not occur frequently enough during training. As a result, word-based models over the "*large dataset"* will minimize rare wordiness and improve the performance of the model. Thus, testing the proposed solution on reasonably large domain dataset is required.
- Test the framework with hybrid of character and word embedding representation to solve "*unknown-word-problem"* in different levels of the proposed framework that uses natural language feature as a vector representation.

References

[1] H. S. Al-Obaidy, and A. Al Heela, "Annotation: An Approach for Building Semantic Web Library", *Applied Mathematics & Information Sciences*, 2012, pp. 133-143.

[2] J. Sangers, F. Frasincar, F. Hogenboom, and V. Chepegin, "Semantic Web Service Discovery Using Natural Language Processing Techniques", *Expert Systems with Applications*, Vol. 40, No. 11, 2013, pp. 4660-4671.

[3] M. Hazman, S. R. El-Beltagy, and A. Rafea, "A Survey of Ontology Learning Approaches," *Int. J. Comput. Appl.*, Vol. 22, No. 9, 2011, pp. 36-43.

[4] A. Browarnik and O. Maimon, "Ontology Learning from Text Departing the Ontology Layer Cake," In *The First International Conference on Big Data, Small Data, Linked Data and Open Data*, 2015, pp. 62-68

[5] K. Ahmad and L. Gillam, "Automatic Ontology Extraction from Unstructured Texts," *On the Move to Meaningful Internet System 2005: Coopis, Doa, and Odbase*, 2005, pp. 1330-1346.

[6] A. Al-arfaj, and A. Al-Salman,"Ontology Construction from Text : Challenges and Trends," *International Journal of Artificial Intellegence and Expert Systems(IJAE)*, Vol. 6.No. 2, 2015, pp. 15-26.

[7] A. Zouaq, D. Gasevic, and M. Hatala, "Unresolved Issues in Ontology Learning - Position Paper -," *Sci. York*, 2011, pp. 52–57.

[8] L. Reeve and H. Han, "A Comparison of Semantic Annotation Systems for Text-Based Web Documents," *Web Semant. Ontol.*, 2006, pp. 165-187.

[9] S. K. Malik, N. Prakash, and S. Rizvi, "Semantic Annotation Framework for Intelligent Information Retrieval using KIM Architecture," *Int. J. Web Semant. Technol.*, Vol. 1, 2010, pp. 12-26.

[10] V. Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. "Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art.," *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 4, No. 1, 2006, p p. 14-28.

[11] L. Reeve and H. Han, "Semantic Annotation for Semantic Social Networks using Community Resources," *AIS SIGSEMIS Bull.*, Vol. 2, No. 3&4, 2005, pp. 52-56.

[12] Z. Ahmed, "Domain Specific Information Extraction for Semantic Annotation," *Physics (College. Park. Md).*, 2009.

[13] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Sci. Am. Mag.*, 2001, pp. 1-4.

[14] T. Slimani, "Semantic Annotation: The Mainstay of Semantic Web," *Int. J. Comput. Appl. Technol. Res.*, Vol. 2, No. 6, 2013, pp. 763-770.

[15] E. On and A. On, "Workshop and Panel on Semantic Annotation.", *International Conference on Generative Approaches to the Lexicon*, 2011.

[16] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna, "MnM: Ontology Driven Tool for Semantic Markup.," In *Proc. Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002)*, 2002, pp. 43-47.

[17] B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov, "KIM-Semantic Annotation Platform", In *The Semantic Web-ISWC*, Springer, Berlin, Heidelberg, pp. 834-849.

[18] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien, "SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation", In *Proceedings of the 12th International Conference on World Wide Web*, ACM, 2003, pp. 178-186.

[19] A. N. El-ghobashy, G. M. Attiya, and H. M. Kelash, "A Proposed Framework for Arabic Semantic Annotation Tool," *Int. J. Comput. Digit. Syst.*, Vol. 51, No. 1, 2014, pp. 45–51.

[20] G. Goethe-universit, A. M. Goethe-universit, M. Dehmer, T. U. Wien, and A. Mehler, "EACL-2006 Web as Corpus," In *Proceeding of 11th Conference of the European Chapter of the Association for Computational Linguistics Proceedings*, 2006.

[21] A. J. Tixier, M. R. Hallowell, B. Rajagopalan, and D. Bowman, "Automation in

Construction Automated Content Analysis for Construction Safety : A Natural Language Processing System to Extract Precursors and Autcomes from Unstructured Injury Reports," *Autom. Constr.*, Vol. 62, 2016, pp. 45–56.

[22] R. Hu, and E. S. Atwell, "A Survey of Machine Learning Approaches to Analysis of Large Corpora", In *Proceedings of SProLaC: Workshop on Shallow Processing of Large Corpora*, Lancaster University, 2003, pp. 45-52.

[23] N. L. Ribeiro, "Extraction of Non-Taxonomic Relations from Texts to Enrich a Basic Ontology", Unpublished Master Thesis, School of Engineering, University of Lisbon, Portugal, 2014.

[24] E. D. Liddy, "Questions to be Asked & Answered as to NLP's Role in Improving Semantic Annotation", In *Proceedings of the 3rd Workshop on Exploiting Semantic Annotations in Information Retrieval*, ACM, 2010, pp. 1-2.

[25] L. Ratinov and J. Turian, "Word Representations : A Simple and General Method for Semi-supervised Learning," In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Association for Computational Linguistics, 2010, pp. 384-394.

[26] T. Mikolov, W. Yih and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations", In *Hlt-naacl*, Vol. 13, 2013, pp. 746-751.

[27] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," In *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process*, 2014, pp. 1532-1543.

[28] E. Agichtein, "Scaling Information Extraction to Large Document Collections", *IEEE Data Eng. Bull*. Vol. 28, No. 4, 2005, pp. 3-10.

[29] S. Sarawagi, "Information Extraction", *Foundations and Trends in Databases,* Vol. 1, No. 3, 2008, pp. 261-377.

[30] L. Reeve, "Integrating Hidden Markov Models into Semantic Web Annotation Platforms", *Technique Report*, 2004.

[31] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna,

"MnM: A Tool for Automatic Support on Semantic Markup," *KMi Technical Report*, No. 133, 2003.

[32] D. C. Wimalasuriya and D. Dou, "Ontology-Based Information Extraction: An Introduction and a Survey of Current Approaches," *J. Inf. Sci.*, Vol. 36, No. 3, 2010, pp. 306-323.

[33] P. V. Bhagat, and P. M. Gourshettiwar, "Survey Paper on Ontology-Based Approaches for Semantic Data Mining", *International Journal on Recent and Innovation Trends in Computing and Communication*, Vol. 3, 2015, pp. 2137-2141.

[34] F. Wu and D. S. Weld, "Open Information Extraction using Wikipedia", In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2010, pp. 118-127.

[35] A. Tan, "Text Mining: The State of the Art and the Challenges", In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases*, Vol. 8, 1999, pp. 65-70.

[36] N. V. Sailaja, L. Padmasree and N. Mangathayaru, "Survey of Text Mining Techniques, Challenges and their Applications", *International Journal of Computer Applications*, Vol. 146, No.11, 2016.

[37] A. Stavrianou, P. Andritsos, and N. Nicoloyannis, "Overview and Semantic Issues of Text Mining," *ACM SIGMOD Rec.*, vol. 36, no. 3, 2007, p. 23.

[38] C. S. Malarkodi, E. Lex, and S. Lalitha Devi, "Named Entity Recognition for the Agricultural Domain", *Research in Computing Science 117*, 2016, pp. 121-132.

[39] G. Lample, "Neural Architectures for Named Entity Recognition," *Arxiv*, 2016, pp. 1–10.

[40] Mikiyas Tadele, "Amharic Named Entity Recognition Using A Hybrid Approach", Unpublished Master Thesis, School of Information Science, Addis Ababa University, 2014.

[41] K. Byrne, "Populating the Semantic Web - Combining Text and Relational Databases as RDF Graphs," *Data Manag.*, pp. 257, 2008.

[42] D. Sánchez, M. Batet, D. Isern, and A. Valls, "Ontology-based Semantic Similarity: A New Feature-based Approach," *Expert Syst. Appl.*, Vol. 39, No. 9, 2012, pp. 7718-7728.

[43] A. M. B. Abdelrahman and A. Kayed, "A Survey on Semantic Similarity Measures between Concepts in Health Domain,", 2015, pp. 204-214.

[44] S. A. Elavarasi, J. Akilandeswari, and K. Menaga. "A Survey on Semantic Similarity Measure", International Journal of Research in Advent Technology, Vol. 2, No. 3, 2014, pp. 389-398.

[45] N. Bahrami, "Computing Semantic Similarity of Documents Based on Semantic Tensors," *American Journal of Computational Mathematics 5*, No. 2, 2015, pp. 204.

[46] L. Piits, "Distributional Hypothesis: Words for 'Human Being' and their Estonian Collocates", *Trames 17*, No. 2, 2013, pp. 141-158.

[47] M. Sahlgren, "The Distributional Hypothesis", *Italian Journal of Linguistics 20*, No. 1, 2008, pp. 33-54.

[48] P. Gupta, R. E. Banchs, and P. Rosso, "International Workshop on Embeddings and Semantics," *SEPLN'15 Workshop*, 2015.

[49] M. Gawich, A. Badr, A. Hegazy, and H. Ismael, "A Survey on Semantic Annotation Tools", *Egyptian Computer Science Journal, ECS*,Vol. 36, No. 2, 2012.

[50] M. Hoffman, F. R. Batch, and D. M. Blei., "Online Learning for Latent Dirichlet Allocation,"In *Advances in Neural Informantion Processing Systems*, 2010, pp. 856–864.

[51] T. K. Landauer, P. W. Foltz and D. Laham,"An Introduction to Latent Semantic Analysis", *Discourse Processes*, Vol. 25, No. 2-3, 1998, pp. 259-284.

[52] A. Saif, N. Omar, M. J. Ab Aziz, U. Z. Zainodin and N. Salim, "Semantic Concept Model using Wikipedia Semantic Features", *Journal of Information Science*, 2017.

[53] T. Grainger, Khalifeh AlJadda, Mohammed Korayem, and Andries Smith, "The Semantic Knowledge Graph: A Compact Auto-generated Model for Real-time Traversal and Ranking of any Relationship within a Domain", In *Data Science and*

Advanced Analytics (DSAA), IEEE International Conference, IEEE, 2016, pp. 420-429.

[54] M. D. Del Castillo and J. I. Serrano, "A Multistrategy Approach for Digital Text Categorization from Imbalanced Documents," *ACM SIGKDD Explor. Newsl.*, Vol. 6, No. 1, 2004, pp. 70-79.

[55] Y. Goldberg and O. Levy, "Word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word Embedding Method", *arXiv preprint arXiv:1402.3722*, 2014.

[56] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *arXiv preprint* arXiv:1607.01759, 2016.

[57] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," In *Proc. Int. Conf. Learn. Represent. (ICLR 2013)*, 2013, pp. 1-12.

[58] G. Sahin and B. Diri, "Extraction of Turkish Semantic Relation Pairs using Corpus Analysis Tool," *Extraction*, Vol. 5, No. 6, 2016, pp. 491-499.

[59] P. Harrington, "Machine Learning in Action," *Greenwich, CT: Manning*, Vol. 5, 2012.

[60] Y. Bengio, I. J. Goodfellow and A. Courville, "Deep Learning," *Nature 521*, 2015, pp. 436-444.

[61] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning : A Survey," *Journal of Artificial Intelligence Research*, Vol. 4, 1996, pp. 237–285.

[62] R. S. Sutton and A. G. Barto, "Reinforcement Learning : An Introduction," *The MIT Press*, 2016.

[63] S. Herculano-Houzel, S. Herculano-houzel, and R. Barton, "The Human Brain in Numbers: a Linearly Scaled-up Primate Brain," *Frontiers in Human Neuroscience*, Vol 3, 2009.

[64] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, 1994.

[65] A. Senior, "Long Short-Term Memory Recurrent Neural Network Architectures for

Large Scale Acoustic Modeling," In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[66] Q. V. Le, "A Tutorial on Deep Learning Part 1: Nonlinear Classifiers and The Backpropagation Algorithm", 2015, pp. 1-18.

[67] A. Java, S. Nirneburg, M. McShane, T. Finin, J. English, and A. Joshi, "Using a Natural Language Understanding System to Generate Semantic Web Content", *International Journal on Semantic Web and Information Systems (IJSWIS)*, Vol. 3, No. 4, 2007, pp. 50-74.

[68] Y. Ding, C. J. van Rijsbergen, I. Ounis, and J. Jose, "Report on ACM SIGIR Workshop on Semantic Web," *SWIR 2003,* SIGIR Forum, Vol. 37, No. 2, 2003, pp. 45-49.

[69] L. Reeve and H. Han, "Survey of Semantic Annotation Platforms," in *Proc. 2005 ACM Symp. Appl. Comput. - SAC '05*, 2005, pp. 1634-1638.

[70] G. P. S. Gosal, "A Survey on Semantic Annotation of Text," *IJARCSSE*, Vol. 5, No. 9, 2015, pp. 54-57.

[71] A. M. Al-Zoghby, A. Sharaf Eldin Ahmed, and T. T. Hamza, "Arabic Semantic Web Applications," *Journal of Emerging Technologies in Web Intelligence*, Vol. 5, No. 1, 2013, pp. 52-69.

[72] N. Babu and Fr. R. Thottupuram, "A Novel Approach for Semantic Similarity Measurement using Spice Ontology and Matchmaking", *International Journal of Engineering Research & Technology (IJERT)*, Vol. 2 Issue 8, 2013, pp. 322-328.

[73] D. Sánchez, D. Isern and M. Millan, "Content Annotation for the Semantic Web: an Automatic Web-based Approach", *Knowledge and Information Systems*, Vol. 27, No. 3, 2011, pp. 393-418.

[74] K. Bontcheva, H. Cunningham, A. Kiryakov and V. Tablan, "Semantic Annotation and Human Language Technology", *Semantic Web Technologies: Trends and Research in Ontology-Based Systems*, 2006, pp. 29-50.

[75] S. Abirami and K.S. Rangasamy, "A Survey: Data Extraction on Web Using Semantic Annotation", *International Journal for Research in Applied Science & Engineering*,

Vol. 3, No. 1, 2015, pp. 177-182.

[76] Q. Rajput, "Ontology based Semantic Annotation of Urdu Language Web Documents", *Procedia Computer Science,* Vol. 35, 2014, pp. 662-670.

[77] K. Balog, J. Dalton, A. Doucet, and Y. Ibrahim, "Report on the Eighth Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR'15)", In *ACM SIGIR Forum,* Vol. 50, No. 1, 2016, pp. 49-57.

[78] H. Bunt, "On the Principles of Interoperable Semantic Annotation", In *Proceedings of the 11th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, 2015, pp. 1-13.

[79] S. Bowers, M. Schildhauer, M. O. Brien, M. Jones, and B. Leinfelder, "A Semantic Annotation Framework for Retrieving and Analyzing Observational Datasets," In *CIKM*, 2010, pp. 31–32.

[80] G. Gao, Y. S. Liu, P. Lin, M. Wang, M. Gu, and J. H. Yong, "BIMTag: Concept-based automatic semantic annotation of online BIM product resources," *Advanced Engineering Informatics*, Elsevier Ltd, 2015.

[81] M. Nagarajan, "Semantic Annotations in Web Services," In *Semantic Web Services, Processes and Applications*, 2006, pp. 35-61.

[82] G. Große-Bölting, C. Nishioka, and A. Scherp, "A Comparison of Different Strategies for Automated Semantic Document Annotation," In *Proceedings of the 8th International Conference on Knowledge Capture*, 2015.

[83] J. Raes, "Designing a Rule-based Annotation System to Enhance Semantic Search on Event-related Articles," 2012.

[84] S. Handschuh, S. Staab and A. Maedche, "CREAM: Creating Relational Metadata with a Component-based, Ontology-driven Annotation Framework", In *Proceedings of the 1st International Conference on Knowledge Capture,* ACM, 2001, pp. 76-83.

[85] H. Hassanzadeh and M. Keyvanpour, "A Machine Learning Based Analytical Framework for Semantic Annotation," *Int. J. Web Semant. Technol.*, Vol. 2, No. 2, 2011, pp. 27-38.

[86] F. Badie,"Concept Representation Analysis in the Context of Human-Machine Interactions", *e-Society*, 2016.

[87] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, and S. Hellmann, "DBpedia-A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia", *Semantic Web*, Vol. 6, No. 2, 2015, pp. 167-195.

[88] F. M. Suchanek, G. Kasneci and G. Weikum, "Yago: A Core of Semantic Knowledge," In *Proceedings of the 16th International Conference on World Wide Web*, ACM, 2007, pp. 697-706.

[89] E. H. Lim, J. Liu and R. Lee, "Knowledge Seeker-Ontology Modelling for Information Search and Management", *Springer-Verlag Berlin An*, 2013.

[90] R. Navigli and P. Velardi, "Ontology Enrichment through Automatic Semantic Annotation of On-Line Glossaries," *Manag. Knowl. a World Networks*, 2006, pp. 126-140.

[91] J. Mccrae, P. Buitelaar, P. Cimiano, T. Declerck, and E. Montiel-, "Lemon : Lexicon Model for Ontologies.", In *Extended Semantic Web Conference, Springer, Berlin, Heidelberg*, 2011, pp. 245-259.

[92] L. Borin, D. Dann, M. Forsberg, and J. P. McCrae, "Representing Swedish Lexical Resources in RDF with Lemon," In *CEUR Workshop Proceedings*, 2014, pp. 329-332.

[93] E. Alatrash, "Using Web Tools for Constructing an Ontology of Different Natural Languages by Emhimed Alatrash," PhD Dissertation, University of Belgrade, 2013.

[94] E. Mäkelä, "Survey of Semantic Search Research," In *Proc. Semin. Knowl. Manag. Semant. web*, 2005, pp. 1–18.

[95] A. Maedche and S. Staab, "Ontology Learning," In *Handbook on ontologie*, Springer Berlin Heidelberg, 2004, pp. 173-190.

[96] A. Maedche and S. Staab, "The Text-To-Onto Ontology Learning Environment", In *Software Demonstration at ICCS-2000-Eight International Conference on Conceptual Structures*, Vol. 38, 2000.

[97] H. Mousavi, D. Kerr, M. R. Iseli, and C. Zaniolo, "OntoGrower: An Unsupervised

Ontology Generator from Free Text," *Tech. Rep.*, No. 4, 2013.

[98] P. Buitelaar, P. Cimiano and B. Magnini, "Ontology Learning from Text: Methods, Evaluation and Applications", *IOS press,*Vol. 123, 2005.

[99] V. Jacobs and P. Monachesi, "Using the Semantics of Prepositions for Ontology Learning", Unpublished Master Thesis, Utrecht University, 2006.

[100] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann, "Modelling Ontology Evaluation and Validation." In *ESWC*, Vol. 4011, 2006, pp. 140-154.

[101] Q. Rajput and S. Haider, "A Comparison of Ontology-based and Reference-set-based Semantic Annotation Frameworks," *Procedia Comput. Sci.*, Vol. 3, 2011, pp. 1535-1540.

[102] J. Kahan and M.-R. Koivunen, "Annotea: an Open RDF Infrastructure for Shared Web Annotations," *Computer Networks*, Vol. 39, No. 5, 2002, pp. 589-608.

[103] A. Kiryakov, B. Popov, and D. Manov, "Automatic Semantic Annotation with KIM," *Semant. Web*, 2004, pp. 1–4.

[104] M. Laclavic, M. Seleng, E. Gatial, Z. Balogh, and L. Hluchy, "ONTEA : Platform for Pattern based Automated Semantic Annotation," *Comput. Informatics*, Vol. 28, 2009, pp. 555-579.

[105] S. Staab, A. Maedche, and S. Handschuh, "Creating Metadata for the Semantic Web - an Annotation Environment and the Human Factor", *Institute AIFB*, 2000, pp 1-25.

[106] S. Albukhitan and T. Helmy, "Automatic Ontology-based Annotation of Food, Nutrition and Health Arabic Web Content," In *Procedia Computer Science*, 2013, Vol. 19, pp. 461–469.

[107] M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna, "MnM: Ontology Driven Semi-automatic and Automatic Support for Semantic Markup," *Knowl. Eng. Knowl. Manag. Methods, Model. Tools Ontol. Semant. Web*, Vol. 2473, 2002, pp. 213-221.

[108] A. Valarakos, R. Valarakos, G. Sigletos, V. Karkaletsis, and G. Paliouras, "A Methodology for Semantically Annotating a Corpus Using a Domain Ontology and

Machine Learning," in *Proceedings of the Recent Advances in Natural Language Processing International Conference (RANLP 2003)*, 2003, pp. 495–499.

[109] H. Zheng, B. Kang, S. Koo, H. Choi, K. Kim, and H. Kim, "A Semantic Annotation Tool to Extract Instances from Korean Web Documents.", In *1st Semantic Authoring and Annotation Workshop of 5th International Semantic Web Conference*, 2006.

[110] D. Maynard, "Multi-Source and Multi-Lingual Information Extraction," *Expert Updat.*, Vol. 6, 2003, pp. 11-16.

[111] M. Lui and T. Baldwin, "Langid.py : An Off-the-shelf Language Identification Tool," In *Proceedings of the ACL 2012 System Demonstration, Association for Computational Linguistics*, 2012, pp. 25-30.

[112] A. M. Dai, C. Olah and Q. V. Le, "Document Embedding with Paragraph Vectors", arXiv preprint arXiv:1507.07998, 2015.

[113] N. V Do, T. PhamNguyen, H. K. Chau and T. T. Huynh, "Improved Semantic Representation and Search Techniques in a Document Retrieval System Design", *J. Adv. Inf. Technol.*, Vol. 6, No. 3, 2015, pp. 146-150.

[114] R. T. Lo, B. He, and I. Ounis, "Automatically Building a Stopword List for an Information Retrieval System", In *Journal of Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, Vol. 5, 2005, pp. 17-24.

[115] P. Pecina, "Collocation Extraction and Theorotical Linguistics," In *Computational and Theoretical Linguistics*, Institute of Formal and Applied Linguistics, 2009.

[116] S. Chung, J. Jun, and D. McLeod, "A Web-Based Novel Term Similarity Framework for Ontology Learning," In *On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA and ODBASE, Proceedings, Part I, Montpellier, France*, 2006, Vol. 4275, pp. 1092-1109.

[117] P. Cimiano, A. Hotho, and S. Staab, "Comparing Conceptual, Divisive and Agglomerative Clustering for Learning Taxonomies from Text", In *Proceedings of the 16th European Conference on Artificial Intelligence*, IOS Press, 2004, pp. 435-439.

[118] B. Desgraupes, "Clustering Indices", *University of Paris Ouest-Lab Modal'X*, 2013.

[119] M. A. Hearst, "Automatic Acquisition of Hyponyms from Large Text Corpora", In *Proceedings of the 14th Conference on Computational Linguistics, Association for Computational Linguistics*, Vol. 2, 1992, pp. 539-545.

[120] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A Neural Probabilistic Language Model," *J. Mach. Learn. Res.*, Vol. 3, 2003, pp. 1137-1155.

[121] Berhanu Mengiste, "Automatic Ontology Learning From Unstructured Text", Unpublished Master Thesis, Department of Computer Science, Addis Ababa University, 2013.

Annexes

Annex A - Related Works and Approaches Comparison

Related Works Comparison

Related Work	IE Tools	IE Method	KB(Ontology)	Annotation	Best Feature(s)	Drawback(s)
KIM	GATE	Pattern based (Rule matching)	KIMO	Semi-Automatic & Automatic	• Extendibility, • Ontology pre-populated with large number of instances	• Challenge to extend the KIM methodology to domain specific ontologies. • Absence of advanced IE
SemTag	Seeker Platform	Taxonomy Label Matching	TAP	Automatic	• Ambiguity resolution • high- performance parallel architecture	• Works only for TAP KB • Provide low accuracy for domain specific documents.
ONTEA	-	-	-	-	• Solve duplicity problem • Generic solution	• High recall and lower precision
MnM	Amilcare	Wrapper Induction	KMi	Semi-Automatic	• Learn extraction rules from training corpus	• Annotation process is very time consuming
PANKOW	PANKOW	Pattern Discovery	User Constructed	Automatic	• Generic solution	• Low success rate
S-CREAM (CREAM)	Amilcare	Rule based/ Wrappers	-	Automatic/ Manual	• Annotating the deep web • Well suited for highly structured web documents	• Difficult to scale
Amaya (Annotea)	-	Based on given parameters	-	Manual	-	• XML and HTML format only and no ontology

Approaches Comparison

Approaches	*Related Work*	*Best Features*	*Drawbacks*
Pattern (Manual, Discovered)	KIM, SemTag, ONTEA, OntoAnnotate, PANKOW	• Rich ontology (KIM) • Genericity (ONTEA & PANKOW) and duplication(ONTEA) • Parallel architecture(SemTag) • Good Precision	• Expensive information extraction • Scalability and adaptation • *No granularity above term* • *Absence of redundancy avoidance* • *No access control* • *All SA requirements consideration*
Machine Learning (Probabilistic, Induction)	MnM(Lazy-NLP), SACML(HMM), SARM(Bayesian & SVM)	• Rule Learning(MnM) • Manual effort reduction	• Classicality • Shallow models • *No granularity above term* • *Absence of redundancy avoidance* • *No access control* • *All SA requirements consideration*
Multi Strategy	MUSE	• Aggregate strengths • Compensate weakness	• Delayed

Annex B - The LangID’s 97 Languages

List of pre-trained 97 languages in langid language detection model are listed here with their ISO 639-1 codes.

af, am, an, ar, as, az, be, bg, bn, br, bs, ca, cs, cy, da, de, dz, el, en, eo, es, et, eu, fa, fi, fo, fr, ga, gl, gu, he, hi, hr, ht, hu, hy, id, is, it, ja, jv, ka, kk, km, kn, ko, ku, ky, la, lb, lo, lt, lv, mg, mk, ml, mn, mr, ms, mt, nb, ne, nl, nn, no, oc, or, pa, pl, ps, pt, qu, ro, ru, rw, se, si, sk, sl, sq, sr, sv, sw, ta, te, th, tl, tr, ug, uk, ur, vi, vo, wa, xh, zh, zu

Annex C - Sample Domain Categorization Training Dataset

[

{"text": "ምክር ቤቱ ከጣሊያን ተቋም ጋር በመተባበር የአቅም ግንባታ ስልጠና ። ንግድምክር ቤት ከተለያዩ የጣሊያን የንግድ ተቋማት ጋር በመተባበር ለሙቶ አባላቱ የአቅም ግንባታ ስልጠና መስጠት የምክር ቤቱ ምክትል ዋና ጸሀፊ በተጀመረው ስልጠና እንደተናገሩት ከኢጣሊያ የስራ ሚኒስቴር ፣ ከቱሪን የንግድና የኢንዱስትሪ ምክር ቤት እና ከ የጣሊያን ንግድ ተቋማት ጋር በመተባበር የንግዱ ሀብረተሰብ አቅም ለማጎልበት የሚያግዙ ስልጠና ተጀምሯል ። በስልጠናው የወጪ ንግድ አመራር ፣ ፋይናንሻል ማኔጅመንት ፣ ፕሮጀክት ማኔጅመንት ፣ ቀረጥና ክፍያዎች ፣ የንግድ ኮንትራቶች እንዲሁም አለማቀፍ ንግድ ነክ ጉዳዮች እንደተካተቱ ። ስልጠናው ከአቅም ግንባታ በተጨማሪ የጣሊያንና የኢትዮጵያን የንግድ ግንኙነት ይበልጥ ለማጠናከር ፣ የጣሊያንና የኢትዮጵያን የንግድ ማህበረሰብ የጋራ የንግድ ልውውጥ ለማሳደግና እንቨስትመንት በአገሪቷ ለማስፋፋት ጠቀሜታ እንደሚኖረው ሃይለመስቀል ። ስልጠናው እስከ የካቲት ወር ማብቂያ እንደሚቆይና በጣሊያንኛና በእንግሊዝኛ ቋንቋዎች እንደሚሰጥ ። ምክር ቤቱ ንግድና እንቨስትመንት በአገሪቷ እንዲስፋፋ ከማበረታታት አንጻር የንግዱን ሀብረተሰብ የንግድ ተቋም የመገንባትና የማስፋፋት አቅም የማጎልበት ፕሮግራም ነድፎ በመንቀሳቀስ ከዋና ጸሀፊው ለመረዳት ። ", "label": "Business"},

.

.

.

{"text": "ለፓርኩ የተፈጥሮ ሀብት ጥበቃ ሚሊየን ብር ተመደበ ። የሰሜን ተራሮች ብሄራዊ ፓርክን የተፈጥሮ ሀብት ለመጠበቅ በኦስትሪያ የልማት ትብብር የተቀናጀ የልማት ፕሮጀክት ሚሊየን ብር ተመደበ ። የፕሮጀክቱን የስራ እቅድ ለማስተዋወቅ ዛሬ በደባርቅ ከተማ በተዘጋጀ የአንድ ቀን የትውውቅ መድረክ የፕሮጀክቱ ስራ አስኪያጅ ተሾመ ሙልዬ ፕሮጀክቱ ገንዘቡን ለመመደብ ያነሳሳው የፓርኩን የተፈጥሮ ሀብት ለዘለቄታው ለመጠበቅና ለጎብኚዎችም ምቹ ሁኔታን ለመፍጠር ነው ። በዚህ መሰረት በተመደበው ገንዘብ በፓርኩ ሊለሙ የሚችሉ ደኖች እንደሚለሙና በደባርቅ ከተማ ባለ አንድ ፎቅ የቱሪስት መርጃ ማእከልና በፓርኩ ውስጥ የፈራረሱት ካርፖች በ መልክ እንደሚገነቡ ስራ አስኪያጁ አስታውቀዋል ። እንዲሁም በፓርኩ አካባቢ የሚኖሩ አርሶ አደሮች ወደ ፓርኩ ገብተው በተፈጥሮ ሀብት የሚያደርሱትን ጉዳት ለመከላከል ስልጠናዎችና የእርሻ ምርቶቻቸውን የሚያሳድጉበት ትምህርት እንደሚሰጥ ተሾመ ለዋልታ እንፎርሜሽን ማእከል ገልጸዋል ። ከአንድ ወር በፊት ስራውን የጀመረውና ጽህፈት ቤቱን በደባርቅ ከተማ ያደረገው ይኸው ፕሮጀክት ባዘጋጀው የትውውቅ መድረክ ከ የሚበልጡ የደባርቅ ፣ ጃን አሞራ ፣ በየዳና አደርቃይ ነዋሪዎችና ከተለያዩ ከቀበሌ ገበሬ ማህበራት የተውጣጡ አርሶ አደሮች ተሳትፈዋል ። ", "label": "Tourism"}

]

Annex D - Sample Indexed Word List

{'ተፈላጊ': 0, 'መደረጉን': 1, 'ተናግረዋል': 2, 'ጉሙዝ': 4, 'የአውሮፓ': 5, 'ከመጨመር': 6, 'ድርጅቶች': 8, 'ገዳማት': 9, 'እንዳይጨፈጭፍ': 10, 'የውስጥ': 177, 'ስለማይደረግላቸው': 11, 'መካሄዱን': 12, 'ወረዳ': 13, 'ዘጠኝ': 16, 'አምባሲ': 18, 'ዘሩ': 17, 'ናቸው': 19, 'አገልጋይ': 21, 'የኢትዮጵያን': 22, 'በመረጃ': 23, 'አእዋፋት': 26, 'ሄራልድ': 25, 'አቤል': 336, 'ትንታዊነት': 27, 'ከ186': 702, 'ትናትና': 28, 'ዋቢ': 180, 'ጥርስ': 30, 'ና': 31, 'ወቅት': 34, 'የሜዳ': 33, 'ባለስልጣን': 35, 'ራስ': 37, 'ተጠንቶ': 337, 'የተባሉ': 820, 'ለማቋቋም': 38, '1993': 39, 'ከስድስት': 40, '406': 43, 'ለማስፋፋት': 44, 'በኩል': 45, 'የሚቻልበት': 46, 'በስኮትላንድ': 47, 'አ.ም': 48, 'ከአምናው': 825, 'በተደረገው': 49, 'ብርሃኑ': 51, 'አቀፍ': 58, 'አባባል': 52, 'ጣሊያን': 53, 'የሚያደንቁ': 55, 'አሊ': 56, 'በመደበው': 60, 'ልዩ': 62, 'ፕሮጀክት': 804, 'የሆሞ': 63, 'ርዝማኔ': 65, 'የተባለውን': 909, 'የደንና': 67, 'ደኑን': 68, 'አመታት': 69, 'ሃለፎም': 70, 'እንቨስትመንትን': 71, 'የነበኙ': 72, 'ደን': 73, '12': 665, 'በእንግሊዝ': 486, 'ወራት': 821, 'ለአውሮፓ': 75, 'ማህበራዊ': 76, 'ለመጥፋት': 78, 'እንዳላቸው': 79, 'መዋቅር': 80, 'ስምምነት': 625, 'ማክሉኪ': 81, 'በዱር': 82, 'በተጀመረው': 672, 'ወንድማገኘሁ': 83, 'የአፈርና': 84, 'ኋይት': 85, 'መደረሱን': 915, 'ስፍራን': 87, 'ማስታወቂያ': 88, 'በ1868': 89, 'የመጀመሪያው': 90, 'ባህል': 91, 'ጥናትና': 92, 'ቢሮው': 93, 'መሬት': 94, 'ዘመን': 95, 'ተብሎ': 96, 'ተመደበ': 97, 'የውጪ': 98, 'ቦታዎችን': 100, 'ሺ': 101, 'አዲስ': 610, 'የተገኙ': 108, 'መሆኑ': 106, 'ጋሻው': 104, 'ኒያላ': 105, '15': 110, 'በቅርስነት': 109, 'የአይጥ': 345, 'አስተዳደር': 111, 'ቤሌ': 112, '\ufeffጋምቤላ': 113, 'መዘግየቱን': 114, 'ሃላፊ': 15, 'መሀመድ': 117, 'በ2': 118, 'ፖርክን': 119, 'ቁጥራቸው': 659, 'የለሚ': 120, 'ንግድ': 973, 'ቴክኖሎጂ': 121, 'መኖሩን': 122, 'ኮሚሽን': 123, 'በሶስት': 124, 'ተራሮች': 125, 'የፕሮጀክት': 126, 'አግኝተው': 20, 'ታሪያና': 128, 'እንዲያገኝ': 129, 'አርባምንጭ': 132, 'የነብር':

.

.

.

'ድልድይ': 468, 'በታላቋ': 319, 'የተቀናጀ': 925, 'ሀብት': 926, 'አካባቢ': 928, 'አምና': 929, 'ክበባትና': 930, 'አምባሲን': 321, 'ውስጥ': 932, 'ሶስት': 130, 'በግሹ': 934, 'ታቦት': 866, 'ትበቃና': 935, 'ሀብረተሰቡ': 936, 'ክፍል': 938, 'አውስትራሊያ': 332, 'ወአእብሃ': 645, '300': 939, 'አስረድተዋል': 940, 'የሚያስተዋውቅ': 941, 'ለመክፈት': 942, 'ክልል': 810, 'ዝንጀሮ': 944, 'አይነቶች': 945, '41': 948, 'ቢቢሲ': 949, 'ቡድን': 950, 'ለንብኚዎች': 324, 'አነስተኛ': 952, 'ከ130': 953, 'ኢኮኖሚ': 954, '186': 957, 'የገዘፈ': 648, 'በቤንሻንጉል': 958, 'በባህል': 518, 'በፓርኩ': 978, 'በክልሉ': 168, 'እንደሚቀየስ': 960, 'መምሪያ': 961, 'ክፍያ': 963, 'ይገኛሉ': 964, 'ፍየል': 965, 'የተሳሳተ': 175, 'ከትናንትበስቲያ': 256, 'በዚህም': 969, 'የነበሩት': 463, 'ኒውስ': 970, 'የጋምቤላ': 971, 'በእንክብካቤ': 530, 'ከ25': 974, 'አሜሪካ': 331, 'ስሙ': 975, 'ጥያቄ': 976, 'ኢጋዘን': 590, 'በህገ': 475, 'ተከልሎ': 979, 'እጅ': 980}

Annex E - Indexed NE Tagsets

{'B-ORG': 1, 'I-PER': 2, 'O': 4, 'I-EV': 3, 'I-M': 17, 'B-M': 6, 'B-DATE': 14, 'I-ORGM': 7, 'B-TIME': 8, 'I-ORG': 13, 'B-ORGM': 9, 'B-CUR': 11, 'B-OC': 16, 'B-EV': 12, 'I-TIME': 10, 'I-LOC': 15, 'B-LOC': 18, 'B-PER': 19, '': 5}

Annex F - Sample IOB Encoded NER Dataset

ጋምቤላ B-LOC
ብሄራዊ O
ፖርክን B-LOC
ለማሻሻል O
110 O
ሺ O
ብር B-CUR
ተመደበ O
። O

የጋምቤላ B-LOC
ክልል O
ብሄራዊ O
ፖርክን B-LOC
በተሻለ O
ሁኔታ O
ለማቋቋምና O
ለነብኚዎች O
ምቹ O
ሁኔታ O
ለመፍጠር O
ለሚካሄደው O
ጥናት O
110 O
ሺ O
ብር O
በመመደብ O
እንቅስቃሴ O
መጀመሩን O
የክልሉ B-LOC
ፕላንና B-ORG
እኮኖሚ I-ORG
ልማት I-ORG
ቢሮ I-ORG
አስታወቀ O
። O

የቢሮው B-LOC
ከፍተኛ O
የፖርክ B-LOC
ባለሙያ O
አቶ O
ወንድማገኘሁ B-PER
ግርማ I-PER
ዛሬ B-DATE
ለዋልታ B-ORG
እንፎርሜሽን B-ORG
ማእከል I-ORG
እንደገለጹት O
ጥናቱ O
የሚካሄደው O
በፖርኩ B-LOC
ያለውን O
የደን B-LOC
ሽፋን O
ደረጃ O
ለማሻሻል O
የዱር B-LOC
እንስሳቱንና B-ORGM
፣ O
አእዋፋቶቹን B-ORGM
ለመለየትና O
መጠናቸውን O
ለማወቅ O
ነው O
። O
ከመስከረም B-DATE
አጋማሽ O
ጀምሮ O
የሚካሄደው O
ይኸው O
ጥናት O
በፖርኩ B-LOC

ክልል B-LOC
ውስጥ O
የሚገኙ O
የመኖሪያ O
ቤቶችና O
፣ O
የእርሻ B-LOC
ቦታዎችን I-LOC
ለማንሳት O
የሚቻልበት O
ሁኔታ O
እንደሚቀየስ O
የጠቆሙት O
ባለሙያው O
በጥናቱም O
በጥናቱም O
ከክልሉ B-LOC
ግብርና B-ORG
ፕላን I-ORG
እኮኖሚ I-ORG
ልማት I-ORG
ቢሮዎች I-ORG
የተውጣጡ O
ከስድስት O
ያላነሱ O
ከፍተኛ O
ባለሙያዎች O
ተሳታፊ O
እንደሚሆኑ O
አስታውቀዋል O
። O
በክልሉ B-LOC
ግብርና B-ORG
ቢሮ I-ORG
የዱር B-LOC
እንስሳት B-ORGM

ልማት O
ቡድን O
ባለሙያ O
አቶ O
ሃለፎም B-PER
መልሶ I-PER
እንዳሉት O
1967 O
በ5ሺ O
61 O
ሄክታር O
መሬት O
ፖርኩ B-LOC
ሲቋቋም O
41 O
አጥቢ B-ORGM
የዱር I-ORGM
እንስሳትና I-ORGM
ሶስት O
ሙቶ O
የሚሆኑ O
የወፍ B-ORGM
ዝርያዎች I-ORGM
እንደነበሩ O
ተቁመው O
፤ O
በልዩልዩ O
ምክንያቶች O
አነስተኛ O
የማይባሉ O
እንስሳት B-ORGM
መሰደዳቸውንና O
መሞታቸውን O
አስረድተዋል O
። O

Annex G - Sample Extracted Single Terms with TFIDF

['ዱር', 0.003878815279072886]	['እንስሳቱ', 0.0006170842489434137]	['በደባርቅ', 0.0004918393567062031]
['ፓርኩ', 0.003438040815541876]	['ዩኔስኮ', 0.0006170842489434137]	['የፊረንጆቹ', 0.0004918393567062031]
['የዱር', 0.0031735761374232707]	['ማስታወቂያና', 0.0006170842489434137]	['መስህቦቹን', 0.0004918393567062031]
['አእዋፍ', 0.002623143235766417]	['ብጹእ', 0.0006170842489434137]	['አገልጋይ', 0.0004918393567062031]
['ቀበሮ', 0.002623143235766417]	['ዝርያ', 0.0005619732296084664]	['አእዋፍት', 0.0004918393567062031]
['ቆርኬ', 0.0022952503312956146]	['አደን', 0.0005289293562372116]	['በኤድንበርግ', 0.0004918393567062031]
['በፓርኩ', 0.0022952503312956146]	['በታዎችን', 0.0005289293562372116]	['በፓርክነት', 0.0004918393567062031]
['ፓርክ', 0.0018512527468302412]	['የድንጋይ', 0.0005289293562372116]	['ግድግዳ', 0.0004918393567062031]
['አጥቢ', 0.0016394645223540103]	['በርሊን', 0.0005289293562372116]	['ከርከሮ', 0.0004918393567062031]
['ስኮትላንድ', 0.0016394645223540103]	['ቁፋሮ', 0.0005289293562372116]	['የቆርኬ', 0.0004918393567062031]
['አብያተ', 0.0014104782832992312]	['የሚኒሊክ', 0.0004918393567062031]	['ህገ-ታቦቱ', 0.0004918393567062031]
['ቅርሶችን', 0.0014104782832992312]	['ሆቴሎችን', 0.0004918393567062031]	['ለቱሪዝም', 0.0004918393567062031]
['ቀበሮዎች', 0.0013115716178832084]	['ኒያላ', 0.0004918393567062031]	['ደንን', 0.0004918393567062031]
['ነብር', 0.0012341684978868274]	['አስመላሽ', 0.0004918393567062031]	['የብጹእ', 0.0004918393567062031]
['ሙዚየም', 0.0012341684978868274]	['አስጎብኝ', 0.0004918393567062031]	['ለቱሪስት', 0.0004918393567062031]
['በመናገሻ', 0.0011476251656478073]	['የፖሊዮአንትሮፖሎጂ', 0.0004918393567062031]	['ብሪትሽ', 0.0004918393567062031]
['ላሊበላ', 0.0011460136051806254]	['አእዋፋትና', 0.0004918393567062031]	['በስዊድን', 0.0004918393567062031]

Annex H - Sample Ontology with Sample Taxonomy Induction Steps

Sample Ontology in OWL Format

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/amharic/ontologies/2017/5/tourism_ontology"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

ontologyIRI="http://www.semanticweb.org/amharic/ontologies/2017/5/tourism_ontology">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Declaration>
      <Class IRI="#gädamə,ʾägäləgayə"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ʾənəsəsatu, masətawäqiyana"/>
   </Declaration>
   <Declaration>
      <Class IRI="#C11"/>
   </Declaration>
   <Declaration>
      <Class IRI="#C12"/>
   </Declaration>
            .
            .
            .
   <Declaration>
      <Class IRI="#[gädamə, ʾägäləgayə], [gobəñəwočə, muziyämočə]"/>
   </Declaration>
   <Declaration>
      <Class IRI="#C28"/>
   </Declaration>
   <Declaration>
      <Class IRI="#C29"/>
   </Declaration>
```

```
   <Declaration>
      <Class IRI="#C3"/>
   </Declaration>
   <Declaration>
      <Class IRI="#[ʾägäru, gobəñi], [ʾənəsəsatu, masətawäqiyana]"/>
   </Declaration>
            .
            .
            .
   <Declaration>
      <Class IRI="#C39"/>
   </Declaration>
   <Declaration>
      <Class IRI="#gobəñəwočə, muziyämočə"/>
   </Declaration>
   <Declaration>
      <Class IRI="#[gädamə,ʾägäləgayə, gobəñəwočə, muziyämočə],[ʾägäru,
gobəñi],[ʾənəsəsatu, masətawäqiyana]"/>
   </Declaration>
   .
   .
   .
   <SubClassOf>
      <Class IRI="#ʾäʾəmočə"/>
      <Class IRI="#C15"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ʾədəsatu"/>
      <Class IRI="#C17"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ʾənəsəsatu"/>
      <Class IRI="#C10"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ʾəsetočačənə"/>
      <Class IRI="#C18"/>
   </SubClassOf>
</Ontology>

<!-- Generated by the OWL API (version 3.4.2) http://owlapi.sourceforge.net -->
```

Sample Taxonomy Induction Steps(Hierarchical Agglomerative Clustering)

Clustering Steps	*Cluster Members (Details)*
C1	ገዳም, አገልጋይ
C2	ፓርኮች, አጥቢ
C3	በመናገሻ, የቱሪዝም
C4	ጎብኝዎች, ሙዚየሞች
C5	ተራራ, የስዊን
C6	ሙዚየሙ, ደኑን
C7	የደብሩ, መስህቦች
C8	ቅርሶቹ, ሰፋሪዎች
C9	አገሩ, ጎብኚ
C10	እንስሳቱ, ማስታወቂያና
C11	በርሊን, አስጎብኝ
C12	የፊረንጆቹ, የዶርዜ
C13	ከዘርፉ, አካባቢን
C14	የባሌ, ግኝቱ
C15	ታቦቱን አእሞች
C16	በወጉ, ስእሎቹ
C17	እድሳቱ, ገላጭ
C18	እሴቶቻችን, በምሁራን
C19	በቡግና, በቦሌ
C20	በስንቅሌ, አራምቤ
C21	ማእከሉና, አሳማ
C22	ለመከለልና, ባለሙያው
C23	ኮሚሽነሩ, የመንበኘት

C24	የዲዛይን, ገቢም
C25	ሳር, [ፓርኮች, አጥቢ]
C26	ወቅዱስ, [በመናገሻ, የቱሪዝም]
C27	[ገዳም, አገልጋይ], [ጉብኝዎች, ሙዚየሞች]
C28	[ተራራ, የስዌን] [ሙዚየሙ, ደኑን]
C29	[የደብሩ, መስህቦች], [ቅርሶቹ, ሰፋሪዎች]
C30	[አገሩ, ጉብኚ], [እንስሳቱ, ማስታወቂያና]
C31	[በርሊን, አስጉብኝ], [የፈረንጆቹ, የዶርዜ]
C32	[ከዘርፉ, አካባቢን], [የባሌ, ግኝቱ]
C33	[ታቦቱን, አእሞች], [በወጉ, ስእሎቹ]
C34	[እድሳቱ, ገላጭ], [እሴቶቻችን, በምሁራን]
C35	[በቡግና, በቦሌ], [በስንቅሌ, አራምቤ]
C36	[ማእከሉና, አሳማ], [ለመከለልና, ባለሙያው]
C37	[ኮሚሽነሩ, የመጎብኘት], [የዲዛይን, ገቢም]
C38	[ሳር, ፓርኮች, አጥቢ], [[ተራራ, የስዌን], [ሙዚየሙ, ደኑን]]
C39	[ወቅዱስ, [በመናገሻ, የቱሪዝም]], [[የደብሩ, መስህቦች], [ቅርሶቹ, ሰፋሪዎች]]
C40	[[ገዳም አገልጋይ, ጉብኝዎች, ሙዚየሞች]], [[አገሩ, ጉብኚ], [እንስሳቱ, ማስታወቂያና]]
C41	[[በርሊን, አስጉብኝ], [የፈረንጆቹ, የዶርዜ]], [[ከዘርፉ, አካባቢን], [የባሌ, ግኝቱ]]
C42	[[ታቦቱን, አእሞች], [በወጉ, ስእሎቹ]], [[እድሳቱ, ገላጭ], [እሴቶቻችን, በምሁራን]]
C43	[[በቡግና, በቦሌ], [በስንቅሌ, አራምቤ]], [[ማእከሉና, አሳማ], [ለመከለልና, ባለሙያው]]
C44	[[ኮሚሽነሩ, የመጎብኘት], [የዲዛይን, ገቢም]], [[[ሳር, [ፓርኮች, አጥቢ]], [ተራራ, የስዌን], [ሙዚየሙ, ደኑን]]
C45	[ወቅዱስ, በመናገሻ, የቱሪዝም], [የደብሩ, መስህቦች], [ቅርሶቹ, ሰፋሪዎች], [ታቦቱን, አእሞች], [በወጉ, ስእሎቹ], [እድሳቱ, ገላጭ], [እሴቶቻችን, በምሁራን]
C46	[ገዳም, አገልጋይ, ጉብኝዎች, ሙዚየሞች], [አገሩ, ጉብኚ], [እንስሳቱ ማስታወቂያና], [በቡግና, በቦሌ, በስንቅሌ, አራምቤ], [ማእከሉና, አሳማ], [ለመከለልና, ባለሙያው]
C47	[ኮሚሽነሩ የመጎብኘት], [የዲዛይን, ገቢም], [ሳር, ፓርኮች, አጥቢ], [ተራራ, የስዌን], [ሙዚየሙ, ደኑን], [ወቅዱስ, በመናገሻ, የቱሪዝም], [የደብሩ, መስህቦች], [ቅርሶቹ, ሰፋሪዎች], [ታቦቱን, አእሞች], [በወጉ, ስእሎቹ], [እድሳቱ, ገላጭ], [እሴቶቻችን, በምሁራን],

C48	[በርሊን, አስጎብኝ], [የፈረንጆቹ, የዶርዜ], [ከዘርፉ, አካባቢን], [የባሌ, ግኝቱ], [ኮሚሽነሩ, የመንብኘት], [የዲዛይን, ገቢም], [ሳር, ፓርኮች, አጥቢ], [ተራራ, የስዌን], [ሙዚየሙ, ደኑን], [ወቅዱስ, በመናገሻ, የቱሪዝም], [የደብሩ, መስህቦች], [ቅርሶቹ, ሰፋሪዎች], [ታቦቱን, አእሞች, በወጉ], ስእሎቹ እድሳቱ, ገላጭ, እሴቶቻችን በምሁራን]
C49	[ገዳም, አገልጋይ, ጎብኝዎች, ሙዚየሞች], [አገሩ, ጎብኚ], [እንስሳቱ, ማስታወቂያና], [በቡግና, በቦሌ, በስንቅሌ, አራምቤ], [ማእከሉና, አሳማ], [ለመከለልና, ባለሙያው], [በርሊን, አስጎብኝ], [የፈረንጆቹ, የዶርዜ], [ከዘርፉ, አካባቢን], [የባሌ, ግኝቱ], [ኮሚሽነሩ, የመንብኘት], [የዲዛይን, ገቢም], [ሳር, ፓርኮች, አጥቢ], [ተራራ, የስዌን], [ሙዚየሙ, ደኑን], [ወቅዱስ, በመናገሻ, የቱሪዝም], [የደብሩ,መስህቦች], [ቅርሶቹ, ሰፋሪዎች], [ታቦቱን, አእሞች, በወጉ], [ስእሎቹ, እድሳቱ, ገላጭ], [እሴቶቻችን, በምሁራን],

Annex I - Sample Source Code

Word Embedding Modeling

```
def trainWord2VecwithWALTANewsCorpus(MODEL_PATH):
    listofSentences=[]
    eachtdoc = io.open('UntaggedTourism.txt', "r", encoding="utf8")
    doctext=eachtdoc.read()
    sentences=[s.strip() for s in re.split(r'[።\?!]' , doctext) if s]
    model = gensim.models.Word2Vec(sentences=listofSentences, size=63, min_count=1, window=3)
    model.save(os.path.join(MODEL_PATH, 'word2vec.model'))
```

Ontology Learning

```
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 20 14:26:49 2017
@author: Kidane
"""
import sys
sys.path.append('D:/Thesis Related/Implementation/Experimentations/TourismDomainOntologyLearner')
from TermExtractor import *
from PairRelationshipMapper import *
from SimilaritiesComputer import *
from NonTaxonomicRelationExtractor import *
from TaxonomicRelationExtractor import *
from TermExtractor import *

def LearnOntology(termExtractionCriteria):
    #Preprocessing and domain concept extraction
    tokinizer_and_candidate_term_selecter()
    distinictTopTermsWriter()
    print("Term Extraction is Done . . .!!!")
    #Concept pairing
    term_to_term_relationship_mapper()
    print("Pair Mapping is Done . . .!!!")
    #Taxonomic induction
    Rterms=load_data()
    similarity= tourism_terms_similarity_with_chiSQUARE(Rterms)
    #similarity=cosineSimilarity()
    hierarchical_agglomerative_clustering(Rterms, similarity)
    print("Taxonomic Induction is Done . . .")
    #Non Taxonomic Relation Extraction
    compute_correlation()
    word2vec_analogy_based_relation()
    print("Non taxonomic relation is Done . . .")
if __name__ == "__main__":
    tec="WI" #WI,WF,POS
    LearnOntology(tec)
```

Semantic Annotation

```
def AnnotateWebDoc():
    model = gensim.models.Word2Vec.load(os.path.join(MODEL_PATH, 'word2vec.model'), mmap='r')
    title="NULL"
    abstract="NULL"
    conclusion="NULL"
    privacyflag=0

    print("> Structure Analyzing")
    print("="*20)
    mytext=convert_pdf_to_txt('D:\Thesis Related\Implementation\Experimentations\DocumentAnalyzer\DocumentToTextConvertor\Documents\TourismNews1.pdf')
    doctext=str(mytext)
    #Extract each paragraph
    alltext=mytext
    mytext=mytext.split('\n\n')
    print(str(len(mytext))+" Structural Units are Detected . . .")
    for paragraph in mytext:
        if(len(paragraph)<10):
            pindex=mytext.index(paragraph)
            del mytext[pindex]

    #Extract title
    temp=mytext[0]
    temp=temp.split()
    if(len(temp)<15):
        title=mytext[0]

    #Extract abstract
    atemp=mytext[1]
    atemp=atemp.split()
    if(atemp[0]=="ረቂቅ"):
        abstract=mytext[1]
    .
    .
    .


    #detect language
    print("> Language Detection")
    print("="*20)
    language=languageDetector(alltext)
    if(language[0]=='am'):
        print("Your document is detected as Amharic document and next phase is domain categorization . . .")
        #proceed to domain categorization
        print("> Domain Categorization")
        print("="*20)
        domain=categorizeDomainOf(alltext)
        print("Your document is categorized under "+domain+" domain")

        redundancyflag=0
        #user and owner privacy checking and privacy flag switching
        if(domain=="Tourism"):
            #Proceed to preprocessing
            print("> Redundancy Checking")
            allsentence=alltext.split("።")
            #redundancyflag=redundancyChecking(allsentence)
            if(redundancyflag==0):
                print("Your document is unannotated document and proceeded into normalization and content extraction . . .")
                #proceed to normalization and content extraction

                #character mapping
                alltext=characteMappings(alltext)

                #*********************************Content Extraction*****************************************
                print("Term Level extraction:")
                #call NER and extract named entities NEs- is_instance_of - Concept
                entities=term_level_extractor(doctext)
                v_entities=termVerifier(entities)
                print(v_entities)
                .
                .
                .
```

```
            paragraph_is_about=[]
            for i in range (0, len(ParagraphContexts)):
                paragraph_is_about.append([ParagraphContexts[i][0], "is_about", ParagraphContexts[i][1]])

            #Content to Concept Mapping
            maps=c2cmapper(entities)

            #Verification
            #level_of_annotation='moderate'

            level_of_annotation='moderate'

            flag=annotation_verification(maps, level_of_annotation)


            #Annotation storage
            if(flag==1):
                rdf_creator(maps)
                rdf_creator(paragraph_is_about)
                print("Successfully Annotated Semantically !!!")
            else:
                print("Unverified annotation that will need change in level of annotation")

        else:
            print("Ontology is not yet modeled for "+domain+" domain")
    else:
        #Reject processing
        print("Is not Amharic Document . . .!!!")

if __name__ == "__main__":
    AnnotateWebDoc()
```

NER Modeling

```
from keras.models import Sequential
from keras.layers.recurrent import LSTM
from keras.layers.core import TimeDistributedDense, Activation
from keras.layers.embeddings import Embedding
import time
def CreateNERModel(global_start_time,X_train, y_train,max_features,
                   embedding_size,hidden_size,out_size,maxlen):
    model = Sequential()
    model.add(Embedding(max_features, embedding_size, input_length=maxlen,
                        mask_zero=True))
    model.add(LSTM(hidden_size, return_sequences=True))
    model.add(TimeDistributedDense(out_size))
    model.add(Activation('softmax'))
    model.compile(loss='categorical_crossentropy', metrics=['accuracy'],
                  optimizer='adam')
    batch_size = 10
    model.fit(X_train, y_train, batch_size=batch_size, nb_epoch=200,
              validation_split=0.10)
    print('Training duration (s) : ', time.time() - global_start_time)
```

Annex J - Ontology Learner Evaluation Questionnaire's Template

ADDIS ABABA UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Questionnaire for Automatic Ontology Learning Evaluation by Linguistic Experts

Dear Participant,

Natural language processing is a theory-motivated range of computational techniques for the automatic analysis and representation of human language like Amharic. This processing requires review of linguistic experts for evaluation and validation of the computational methods and results. In our case, we focus on how to conceptualize real world entities and create domain knowledge automatically from unstructured Amharic news texts of tourism domain.

Therefore, this is to kindly seeking your linguistic evaluation, decision, comments and suggestions on the result of our knowledge learning model that has to be checked to assess the correctness and validity of the modeled knowledge of the tourism domain from linguistic point of view.

Department: *Computer Science*

Name: [redacted]

Phone: [redacted]

Expert Profile:

The aim of the profile is to make a summary of the linguistic experts evaluated our research experimentation result. Information of the individual linguistic expert will be kept anonymous and used for the research purpose **ONLY**.

Position: _____

Linguistic Experience: _____

THANK YOU FOR YOUR PARTICIPATION!!!

1. Single Term Extraction Experimental Result

Here the focus is to get your judgement on candidate terms and instances relevancy as relevant and irrelevant candidate of the domain ontology which are used to represent/conceptualize tourism domain knowledge.

Criteria 1: POS based Extracted Terms

S.No.	Extracted Domain Terms	Expert Decision	
		Relevant	Irrelevant
1.			

Criteria 2: Wikipedia Frequency based Extracted Terms

S.No.	Extracted Domain Terms	Expert Decision	
		Relevant	Irrelevant
1.			

Criteria 3: Wikipedia Infobox based Extracted Terms

S.No.	Extracted Domain Terms	Expert Decision	
		Relevant	Irrelevant
1.			

Comments and suggestions on the extracted single term extraction:

_____.

2. Multi Term Extraction Experimentation Result

Please judge the correctness of the multi word expressions provided in the following table as correct and incorrect in the space provided here below.

2.1. How many number of words are valid for multi term expression in Amharic _____?

Criteria 1: Chi-square based bigram terms extraction

S.No.	Extracted Bigram and Trigram Domain Terms	Expert Decision	
		Correct	Incorrect
1.			

Criteria 2: Google's word2phrase based bigram terms extraction

S.No.	Extracted Bigram and Trigram Domain Terms	Expert Decision	
		Correct	Incorrect
1.			

Comment and suggestion on the extracted multi term extraction:

_____.

3. Taxonomic Clustering Experimentation Result

Each concept representative terms extracted in number 1 and 2 of this questionnaire have to be represented as a set of terms hierarchically or taxonomically in ontology typically includes a hierarchical is-a relation between concepts and cluster of concepts. Although various other relations between concepts may also defined, is-a relationship is often particularly an important type of relationship between concept. Please judge the clustering quality as very good, good, fair and bad in the space provided here below.

Clustering Iterations	Created Cluster	Cluster Label	Expert Decision			
			Very Good	Good	Fair	Bad
1						

Comment and suggestion on the clustering:

_____.

4. Non taxonomic Relation Experimentation Result

The other semantic relationship between concepts may also defined using concept pair to verb correlation statistical evidences that related as *domain-label-range* relation. The obtained result from the experiment is presented in the following table. Please judge the labeled relation as correct, acceptable, and incorrect in the space provided here below.

S.NO.	Domain	Label	Range	Expert Decision		
				Correct	Acceptable	Incorrect
1.						

Comments and suggestions on the created non taxonomic relation:

_____.

THANK YOU AGAIN FOR YOUR PARTICIPATION!!

Annex K - Manual Semantic Annotation Questionnaire's Template

ADDIS ABABA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Questionnaire for News Articles Manual Annotation

Dear Participant,

Natural language processing is a theory-motivated range of computational techniques for the automatic analysis and representation of human languages like Amharic. This processing requires review of the linguistic experts for the evaluation and validation of the computational methods and results. In our case, we focus on how to annotate Amharic documents with semantic descriptions. Therefore, this is to kindly seeking your help for the selection and annotation of representative terms of the news articles with the term's that convey the meaning of the news articles.

Purpose

The manual annotation is to be used as a gold standard in order to evaluate the proposed annotator for Amharic documents using tourism domain news.

Department: *Computer Science*

Name: [redacted]

Phone: [redacted]

Expert Profile:

The aim of the profile is to make a summary of the linguistic annotated the dataset used to evaluate our proposed solution for Amharic document annotation. Information about the individual linguistic expert kept anonymous and used for the research purpose ***ONLY.***

Name (Optional): ______________________________

Position (Optional): ______________________________

Linguistic Experience: ______________________________

Signature: ______________________________

THANK YOU FOR YOUR PARTICIPATION!!!

1. [illegible]

[illegible] ::

2. [illegible] ::

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
1.				

[illegible]:

____

____

____

____

****** [illegible] ! ******

Declaration Sheet

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

Declared by:

Name: Kidane Woldemariyam

Signature: _____

Date: _____

Confirmed by advisor:

Name: Fekade Getahun (PhD)

Signature: _____

Date: _____