Universal Networking Language Based Question Answering System for Information Retrieval of Punjabi Language

A Thesis
Submitted in the fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

Submitted by

Vaibhav Agarwal

(Registration No. 951303003)

Under the supervision of

Dr. Parteek Kumar

Associate Professor

Computer Science and Engineering Department,
Thapar Institute of Engineering and Technology, Patiala

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

February 2019

Certificate

I hereby certify that the work which is being submitted in this thesis entitled "Universal Networking Language Based Question Answering System for Information Retrieval of Punjabi Language", in fulfillment of the requirements for the award of the degree of DOCTOR OF PHILOSOPHY submitted in Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Parteek Kumar and refers work of other researchers which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of degree in any other University.

Vaishu 20/2/2019.

(Vaibhav Agarwal)

Regd. No. 951303003

Defrendig :- 22/oct/2019

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge and belief.

Parteek Kumar 20.2.2019

(Parteek Kumar)

Associate Professor, Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala-147004 (INDIA)

Supervisor

Acknowledgements

With the grace of God, it is very special moment to acknowledge the contributions of those who supported me during the journey to achieve this task. This research work is the outcome of very valuable guidance from my supervisor Dr. Parteek Kumar. He has always been there to guide me towards the right direction whenever I got stuck with the ideas. I want to thank him for his valuable evenings and week-ends that he spent with me in finalizing the writings of this thesis work. I will always remain indebted to him for showing me the path. I also want to thank Mrs. Kumar to provide a conducive environment in her home during discussions on this proposed work.

I express my gratitude to the Doctoral Committee for monitoring the progress and providing valuable suggestions for improvement of my research work. I feel very proud and thankful to the Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala for providing all the resources for a good research work. I am also very thankful to Dr. Maninder Singh (Head CSE department, TIET, Patiala) and all my colleagues for their constant support and cooperation during my research work.

I feel very proud from the core of my heart in thanking my ex-employer MAQ SOFTWARE. Because of the working culture in this MNC and the number of projects in which I got an oppourtunity to work on different technologies helped me in sharpening my technical and management skills.

I take this opportunity to say thanks to my family members, who are always there to support me morally and emotionally. With the grace of God, this work provides me the opportunity to make my parents, brother, and wife proud.

At the end I wanted to thanks all my friends Gourav Garg, Ruchit Bhatt, Abhishek Singh, Saurabh Jain, Desh Deepak Pathak, Jitendra Khedar, Anurag Diwedi who have been stress busters for me.

(Vaibhav Agarwal)

Abstract

During the last couple of years, in the field of Natural Language Processing, QA (*i.e.*, Question Answering) systems and UNL have been an area of immense research among researchers. This thesis is devoted to develop a language independent Question Answering System (*i.e.*, QAS) for Punjabi language based on UNL. The complete abstract and chapter-wise summary of thesis is given as follows.

First chapter provides an overview of QA systems including its need, challenges, classification, and significance of UNL for QA systems. This chapter also highlights the basics of UNL and its building blocks. It highlights the differences and advantages of UNL over other traditional approaches. Unlike the traditional approaches and techniques in natural language processing, scope and use of UNL is not limited to one domain. How UNL can be exploited for other NLP tasks has been covered under this chapter. The working principle of UNL for QA system has been explained in this chapter.

Based on analysis and survey of various QA systems, several gaps were identified and objectives were set in order to fill those gaps. The gaps that were identified are: non availability of QA system for Punjabi language, lack of integration with other QA system, lack of support for multilingualism, lack of integration of different NLP applications, and complexity. In order to address these gaps, the core objectives of this research proposal has been framed and accomplished. This chapter highlights the contributions to this thesis.

Second chapter of this thesis is background theory and literature review. It focuses on the analysis and study done of various QA systems, comparison of some important QA systems, and study of various UNL based activities. In this chapter, the literature review done has been documented. The complete literature review has been divided into two parts, *i.e.*, research activities in question answering system, and research activities in UNL. The various parameters on the basis of which the comparison of various QA systems has been done are: corpus used for testing, evaluation metrics, evaluation metric's value, domain, is working algorithms of all components explained, is question answering system available online, does question answering system supports multiple language, can question answering system be extended to support

other foreign languages, can question answering system integrate other NLP applications, and is source code available for the given question answering system.

Some of the important QAS which have been covered in this subsection are MMQA developed by Gupta *et al.* (2018), EARL developed by Mohnish *et al*. (2018), CQASMD developed by Feng *et al.* (2018), QA4IE developed by Lin *et al*. (2018), Web Shodh developed by Chandu *et al*. (2017), automatic question answering system for the Arabic Quran developed by Mohamed (2017), FelisCatusZero proposed by Kotaro *et al*. (2017), a semantic network theory to build up intelligent answering system based on remote service framework designed by Xiaoyi *et al*. (2017), a question answering system supporting vector machine method for hadith domain developed by Nabeel and Saidah (2017), a Wikipedia Based Essay Question Answering System for University Entrance Examination proposed by Takaaki *et al*. (2017) , an automated QA system using a hybrid approach proposed by Kwong and Chih (2017), a design of intelligent tourism QA system based on semantic web proposed by Hua and Shi-zheng (2017), an automatic web-based question answering system for e-learning developed by Waheeb and Babu (2017), a public platform for developing language-independent applications developed and tested by Agarwal and Kumar (2017), an information retrieval system using UNL by Goel (2016), a multilingual cross-domain client application prototype for UNLization and NLization for NLP applications developed by Agarwal and Kumar (2016), a modular QA system pipeline called as YodaQA developed by Baudiš (2015), use of vector space model in Question Answering System proposed by Hartawan and Suhartono (2015), a Long Short-Term Memory Model for Answer Sentence Selection in Question Answering proposed by Wang and Nyberg (2015), a Question Answering system using learning knowledge graphs through conversational dialog proposed by Ben *et al.* (2015), a Hybrid QA system (ISOF) over linked data and text data developed by Park *et al*. (2015), an Answer Selection for community Question Answering (QCRI) developed by Nicosia *et al*. (2015), CICBUAPnlp which is a Graph-Based Approach for Answer Selection in Community Question Answering Task proposed by Helena *et al*. (2015), CASIA@V2 which is an MLN-based Question Answering System over Linked Data developed by Shizhu *et al*. (2014), Al-Bayan which is an Arabic QA system for the Holy Quran developed by Heba *et al*. (2014), Forst which is a QA system using basic element at NTCIR-11 QA-Lab Task developed by Kotaro *et al*. (2014), a Knowledge-

Based QA as Machine Translation developed by Junwei *et al*. (2014), CMU Multiple-choice Question Answering System developed by Di *et al*. (2014), a natural language QA system in Malayalam using domain dependent document collection as repository developed by Pragisha and Reghuraj (2014), a QA system called as Watsonsim using the Indri, Lucene, Bing and Google search engines, Apache UIMA, Open NLP, and Weka developed by Sean *et al*. (2014), architecture of a Question-Answering System for a Specific Repository of documents proposed by Manuel and Riofrio (2010), a 'LOOK4' system using Universal Words (UWs) to enhance web search results proposed by Avetisyan and Avetisyan (2010).

The subsection of research activities in UNL highlights the major research activities in UNL. Some of the important research activities in UNL covered in this subsection are English to Tamil machine translation system using UNL by Sridhar *et al.* (2016), Development of dictionary entries of Bangla repetition words to integrate them into UNL by Roy *et al.* (2016), Formation of word dictionary of Bangla vowel ended roots for first person for UNL by Ali *et al.* (2015), Multilingual acquiring of e-content definition based on UNL by Sathiyamurthy *et al.* (2015), creation of a Language-Independent Discourse Parser using UNL by Navaneethakrishnan *et al.* (2015), a new approach of solving semantic ambiguity problem of Bangla Root words using UNL proposed by Mridha *et al.* (2014), Development of Analysis Module for Punjabi language by Agarwal (2013), Development of Generation Module for Punjabi language by Verma and Bhatia (2013); Singh and Bhatia (2013), Development of Punjabi EnConverter and DeConverter by Kumar and Sharma (2012, 2013), Enhancement of web search results through UNL by Avetisyan and Avetisyan (2010), Development of English EnConverter and DeConverter by Jain and Damani (2009), Multilingual search engine with the use of UNL proposed by Karande (2007), Development of Arabic DeConversion system by Adly and Alansary (2009), and Language-Independent Universal Digital Library within UNL framework proposed by Alansary *et al*. (2006).

Having done an exhaustive survey on various QA systems, it has been observed that the proposed UNL based (online available) QA system will definitely be a major step in NLP and removing the language barrier.

In **third chapter** of this thesis, architecture and working of the proposed question answering system has been discussed. Requirements and functionalities of all these

architecture components have been documented. This chapter also describes the interface of the developed question answering system along with the technology and programming language used. Towards the last of this chapter, data structures, *i.e.*, JSON objects which are formed during the initial phase of the question answering system have also been highlighted. These data structures are further used by different modules of the developed question answering system to give the final result. This chapter sets the expectation, understanding, high-level overview of all things that would be covered in subsequent chapters of this thesis. It basically lays the foundation of the subsequent chapters and thesis organization. The architecture/working of the proposed QA system has been divided into the phases *viz.* UNLization phase (Analysis Module), Preprocessing and Crawling phase (UNL Crawler), Optimizing and Ranking phase (Optimizer), and NLization phase (Generation Module).

Analysis module of the proposed question answering system invokes UNLization module of the source natural language to UNLize the question asked by the user and the corpus. The UNL corpus forms the UNL repository. UNL crawler crawls UNL of the question and UNL corpus to find the answer. Optimizer analyses the answer given by UNL crawler and gives ranking to it. This answer is converted to UNL and is given as an input to generation module of the proposed question answering system which invokes the NLization module of the target natural language to get the required final answer in this target natural language.

In the proposed UNL based question answering system user can ask a question in any natural language and can get the output in any natural language. This is possible because the proposed system converts natural language to UNL and works on this generated UNL. Similarly, the output given by the optimizer is converted to UNL and given to the generation module (discussed in Chapter 6) which gives final output in the target natural language. This feature of the proposed question answering system makes it natural language independent. However, in order to UNLize the corpus and question asked by the user, UNLization module of the source language (in which question is being asked) needs to be developed and invoked by the analysis module. UNLization is done with the help of online tool IAN (*i.e.*, Interactive ANalyzer) whereas NLization is done by using EUGENE (*i.e.*, dEep-to-sUrface GENErator). Both IAN and EUGENE have been developed by UNDL foundation available at http://dev.undlfoundation.org/analysis/login.jsp.

The **fourth chapter** focuses on the UNLization process and results of the UNLization module for the Punjabi language. This chapter also illustrates how the UNLization module of the source natural language is invoked by the analysis module of the proposed system.

In starting of this chapter, the framework of IAN tool which is used for UNLization has been explained. This chapter gives the idea of phases of the UNLization process. Each of these phases has been explained in this chapter with the help of example sentences. Documentation regarding UNLization artifacts like Normalization, TRules, DRules, and Analysis Grammar *etc.* has been done in this chapter. This chapter introduces the X-Bar theory and its need in UNLization. How UNLization is done using this X-Bar theory has been illustrated in this chapter. The working of the developed UNLization module using IAN has been illustrated with the help of example sentences. Towards the end of this section, a brief introduction about EUGENE is also given so that the configuration and invoking of UNLization and NLization modules should be clear to the reader.

This chapter highlights the use of UNLization and NLization module from the developed question answering system perspective and gives details about how analysis and generation modules of the proposed question answering system can be used to invoke UNLization and NLization module. It gives details about the prerequisites and other necessary steps for configuration.

After explaining about IAN, analysis, and UNLization modules, the evaluation metrics and results of the UNLization module of Punjabi language have been described in this chapter. The same section also gives details of the corpora that are used for testing the UNLization module. This chapter highlights the types of errors which exist in the UNLization modules due to which F-Measure becomes less than 1. Details of how to calculate those errors have also been explained in this chapter.

The achievements and contributions of the developed UNLization module for the Punjabi language have been highlighted in this chapter. The UNLization module had been submitted for UNL Olympiad II, III, and IV conducted by UNDL foundation in July 2013, March 2014, and November 2014 for UC-A1, UGO-A1, and AESOP-A1 respectively. UC-A1, UGO-A1, and AESOP-A1 are the corpora provided by UNDL foundation. The language selected for the proposed question answering system *i.e.*, Punjabi, had been selected in top 5 (Based upon the F-Measures) UNLization

grammars for Olympiad III, and Olympiad IV while it was selected in top 10 best grammars for Olympiad II. The current updated UNLization module for the Punjabi language has 1798 dictionary entries, 24 NRules, 50 DRules, and 1259 TRules.

In **fifth chapter**, the detailed information of UNL crawler and optimizer has been documented. Initially, in this chapter the concept of preprocessing has been explained with the help of example sentence. This chapter also documents the crawling, optimizing, and ranking processes. The pseudocodes of crawling, optimizing, and ranking have been explained in this chapter.

In the proposed system, the questions asked by the user from the developed question answering system have been categorized into three different types, *i.e.*, missing type question, polar (Yes/No type) question, and non-missing type question. This chapter explains these question types with the help of example sentences.

After optimizing phase, in the ranking phase, optimizer gives rank 1, rank 2, rank 3, rank 4, or rank 5 to the answer found by the UNL crawler. In this chapter, the ranking mechanism used by the optimizer has been explained for every case with the help of example sentences. If all the information in question's UNL is present in corpus's UNL and answer given by UNL crawler is not blank, then optimizer says that "*it's a perfect match. Your answer is:*". If not all the information in question's UNL is present in corpus's UNL and answer is not blank, then optimizer reports that "*It's ALMOST a perfect match. We cannot find some information in Database*". If there is a mismatch between the information in question's UNL and information present in corpus's UNL and answer given by UNL crawler is not blank, then optimizer reports that "*It's a partial match. The most probable answer is*". If all the information in question's UNL is present in corpus's UNL but the answer given by UNL crawler is blank, then optimizer reports that "*Your question is correct but we are sorry because our database does not contain sufficient information to answer this*". If answer given by UNL crawler is blank, then optimizer reports "*Cannot find the answer*".

Optimizer also makes sure that if user didn't ask any question, then no further processing is done and it stops the execution and searching/finding process immediately by alerting the user that "*Sorry we cannot find any question*".

In **sixth chapter**, the detailed information of generation and NLization modules for the Punjabi language has been documented. This chapter highlights the framework of EUGENE tool which is used for NLization. The working of the developed NLization

module using EUGENE has been illustrated with the help of example sentences.

This chapter highlights the use of NLization module from the developed question answering system perspective and gives details about how generation module of the proposed question answering system can be used to invoke NLization module of target natural language.

After explaining about EUGENE, generation and NLization modules, the state of art of NLization module of Punjabi language has been discussed in this chapter. The achievements and contributions of the developed NLization module for the Punjabi language have been highlighted in this chapter. The language selected for the proposed QA System, *i.e.*, Punjabi, has been selected in top 5 (Based upon the F-Measures) NLization grammars for Olympiad III, and Olympiad IV conducted by UNDL Foundation in March 2014, November 2014. The current updated NLization module for the Punjabi language has 704 UWs, 462 TRules, and 10 inflectional paradigms.

In **seventh chapter**, the detailed information of experimentation and evaluation of the proposed question answering system has been documented. This chapter gives the details of corpora used for testing the developed question answering system. This chapter introduces the evaluation metrics *viz*. Conciseness, Relevance, Correctness, Precision, Recall, and F-Measure for evaluating the developed question answering system. The details regarding the total number of questions and their types have been highlighted in this chapter. The Conciseness, Relevance, Correctness, Precision, Recall, and F-Measure of the developed question answering system came out to be 89.5%, 86.4%, 100%, 86.4%, 100%, and 92.7% respectively.

The analysis/comparison of the developed question answering system on the basis of F-Measure/accuracy has been done with different question answering systems. The error analysis of Relevance and Conciseness has also been performed and explained in this chapter. The testing methodology and example questions along with their answers have also been listed in this chapter.

Chapter eight presents the conclusion and future scope of this research work. Limitations of the developed QA system have also been documented in this chapter. As a part of this PhD thesis, various gaps were identified in the existing question answering systems and objectives were framed to address these identified gaps. A framework for UNL based question answering system has been proposed to meet all

the objectives. Based on this framework, the proposed question answering system has been developed and tested for Punjabi language. The Punjabi language resources for UNLization and NLization modules have been created. A public platform for developing UNL based language-independent applications has been developed and tested. The analysis and generation modules of this platform invoke IAN and EUGENE module of the source and target natural language (Punjabi in this case) for UNLization and NLization respectively. IAN/UNLization and EUGENE/NLization modules have been developed for Punjabi natural language.

UNDL foundation had conducted a series of Olympiads over the years. The UNLization/ IAN module had been submitted for UNL Olympiad II, III, and IV conducted by UNDL Foundation in July 2013, March 2014, and November 2014 for UC-A1, UGO-A1, and AESOP-A1 respectively. The language selected for the proposed question answering system *i.e.*, Punjabi, had been selected in top 5 (Based upon the F-Measures) UNLization grammars for Olympiad III, and Olympiad IV while it was selected in top 10 best grammars for Olympiad II. The current updated UNLization/IAN module for the Punjabi language has 1798 dictionary entries, 24 NRules, 50 DRules, and 1259 TRules.

The language selected for the proposed QA system, *i.e.*, Punjabi, has been selected in top 5 (Based upon the F-Measures) NLization grammars for Olympiad III, and Olympiad IV conducted by UNDL Foundation in March 2014, November 2014. The current updated NLization/EUGENE Module for the Punjabi language has 704 UWs, 462 TRules, and 10 inflectional paradigms.

The Conciseness, Relevance, Correctness, Precision, Recall, and F-Measure of the developed question answering system came out to be 89.5%, 86.4%, 100%, 86.4%, 100%, and 92.7% respectively.

The ‘Conciseness’, ‘Relevance’, ‘Precision’, and ‘F-Measure’ metric values of the proposed QA system can be improved. IAN and EUGENE modules can be enriched for Punjabi language so that their F-Measure can be increased. Since a public platform for developing language-independent applications has been developed and tested, therefore other NLP applications which are UNL based like sentiment analysis, text summarization, machine translation *etc.* can be developed and integrated with this. The developed system can be extended to support the feature to upload the UNL corpus by the user so that questions can be asked by the worldwide audience.

List of Figures

List of Tables

List of Pseudocodes

Certifications in UNL

Contents

Chapter 1

Introduction

Information Retrieval (IR) is an activity of getting information relevant to our need from a collection of information resources. Current information retrieval systems are based on keywords in which the result is in the form of the document list. The number of retrieved documents is large. The user has to traverse through these documents to get information of interest. The user searches these documents step by step to find the correct answer. Sometimes, it is difficult to find the correct or relevant answer from the searched keywords. Moreover, an average user seeking an answer to the question searches very few documents. Information retrieval systems broadly have two operations, these are Input and Output (voluminous). Input is the list of keywords given to a search engine, and output is the list of full-length documents. The basic working of IR systems is explained below with the help of Figure 1.1.

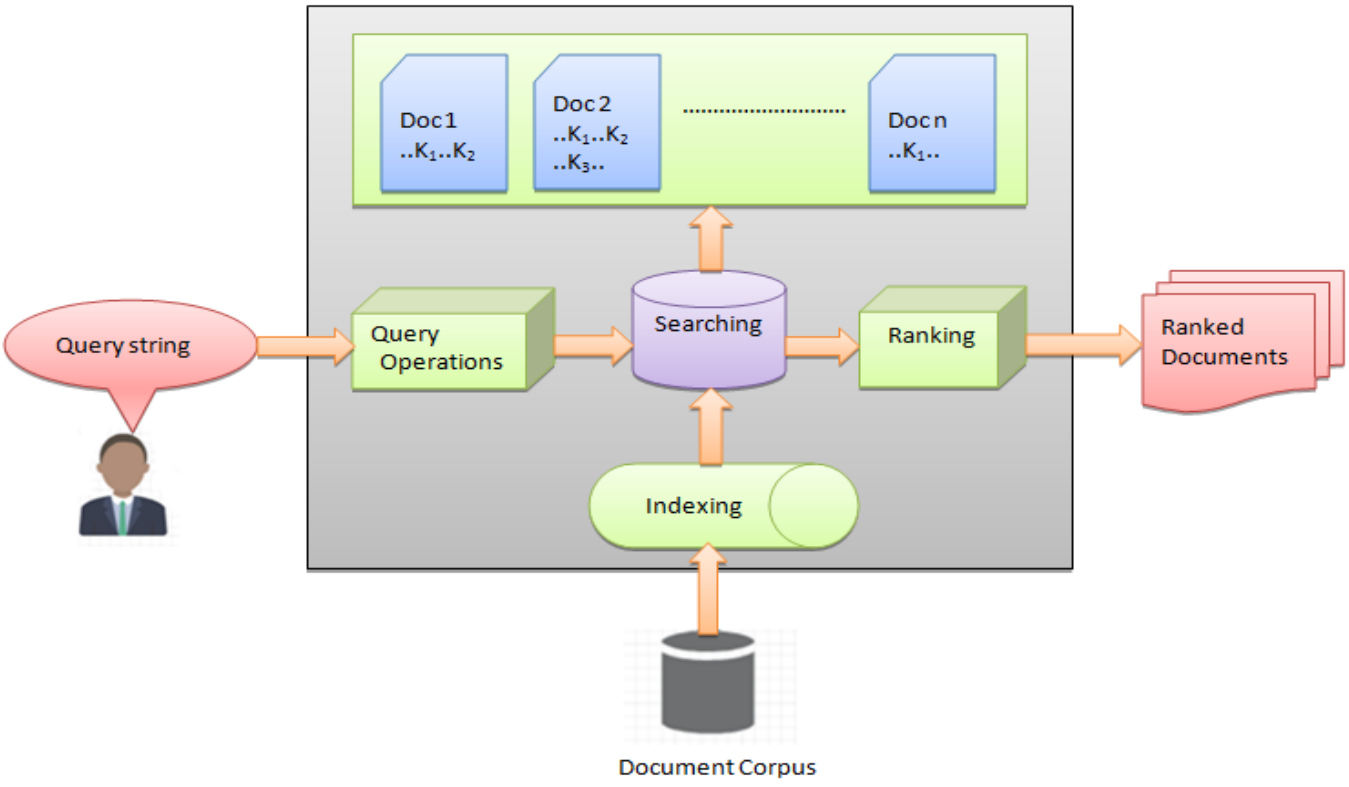


Figure 1.1: Information Retrieval Systems

Assume that a query string contains keywords K_1, K_2, and K_3. In information retrieval systems some query operations, if required, are done on the query string. Searching is done on indexed document corpus based on this restructured query. A set of relevant

documents are retrieved by IR systems and are ranked as per the ranking algorithm as shown in Figure 1.1. As the search is tedious, it demotivates the user and he/she gets tired if the documents do not contain the content which they are searching for. In this scenario, Question Answering (QA) systems stand as an implementation for information retrieval systems.

1.1 Introduction to Question Answering Systems (QASs)

Information extraction systems extract a number of opinions acting as a source for both novel QA systems and commonsense knowledge [112]. These systems help in identifying a set of relations from the text which support in question answering [139]. QA engines attempt to let you ask your question the way you would normally ask in natural language. These queries are more specific than short keyword queries. Also, it is very useful for inexperienced search users. The two main tasks of QASs are Input, *i.e*., Natural Language (NL) questions/expressions, and Output, *i.e.*, list of appropriate short answers. Question answering systems provide intuitive information access. Question answering can be considered as a real use of natural language processing which it exploits completely. The aim of question answering systems is to answer NL questions from any type of document. These documents can be classified into three categories as given below [45].

a) **Documents**: These can be structured or unstructured such as web pages, multimedia documents consisting of errors, contradictions and inconsistencies. Structured documents are used by database question answering systems while unstructured documents are used by web based QAS.
b) **Domain**: This can further be classified into open domain like newspaper articles and close domain like aerospatiale domain, medical domain, *etc*.
c) **NL questions**: These contain either factoid questions like who, when, where, how much, *etc.*, or definition questions like how and why.

1.2 Components of Question Answering Systems

A typical QA system consists of three distinct modules, each of which has a core component beside other supplementary components [11]. These modules are as follows.

- Query processing module, whose purpose is to classify the question.
- Document processing module, whose purpose is retrieval of information.

- Answer processing module, whose task is to extract the answer.

Question processing is the module that identifies the question's focus, classifies the question type, expected answer type derivation, and reformulation of question into semantically equivalent multiple questions.

Reformulating the question into similar meaning questions is also called as query expansion and it helps in boosting up the recall of a particular information retrieval system. Information retrieval system's recall is very important for question answering because when no correct candidate answers are present in a document, then for finding an answer no further processing could be carried out. Ranking and precision of candidate passages can also affect question answering performance in the information retrieval phase.

The final component is answer extraction in question answering system, which is a differentiating feature between the usual sense of text retrieval systems and question answering systems. Answer extraction method becomes a decisive and influential factor on QAS for extracting the final results. Therefore, in the question answering system, the answer extraction technology is deemed to be a module [11].

1.3 Classification of Question Answering Systems

Question answering systems has been classified into several types based on many criteria as identified by Amit and Sajay (2016) [13]. These are as follows:

- Application Domains
- Types of questions asked by users
- Types of analysis done on source documents and users' questions
- Type of data consulted in the data source
- Characteristics of data source
- Types of representation of question data and its matching function to generate candidate answers
- Type of techniques used for retrieving answers
- The forms of answer generated by QAS

QASs can be further classified into different categories on the basis of techniques used in QASs as given by Amit and Sajay (2016) [13]. The different techniques for generating answers are good in their respective scenario. The scenario refers to the complexity of questions, data sources, and answers desired by users. These techniques

are briefly described as follows.

- **QASs using data mining techniques**: In this category, factual data is searched and short answers using bag of word model are generated from database.
- **QASs using information retrieval techniques**: In this case, factual information in text documents is searched.
- **QASs using natural language processing and understanding techniques**: In this case, subjective or objective information is searched.
- **QASs using knowledge retrieval**: In this case, a search is performed for understanding and creating knowledge. Universal Networking Language (UNL) is one such approach which is based on knowledge retrieval. Section 1.4, 1.5, and 1.6 cover the introduction, history, and building blocks of UNL in detail. In this thesis, a QA system has been proposed by using UNL based technique.

1.4 Introduction to UNL

UNL, unlike a natural language, is a non-spoken artificial language to summarize, describe, represent, and store information in a natural language independent format [127]. One way of introducing the UNL is to present it as the "language of computers", which is different in nature from the concept of "computer language" such as JAVA, BASIC, C++ *etc* [149]. In UNL there are two processes, *i.e.*, UNLization and NLization. The process of analyzing, representing, and mapping the information present in any natural language resource into UNL (in the form of UNL graph) is called UNLization whereas the process of realizing, manifesting, and generating a natural language resource from a UNL graph is called NLization [127]. Both the processes are not dependent on each other [4]. A typical UNL system is shown in Figure 1.2.

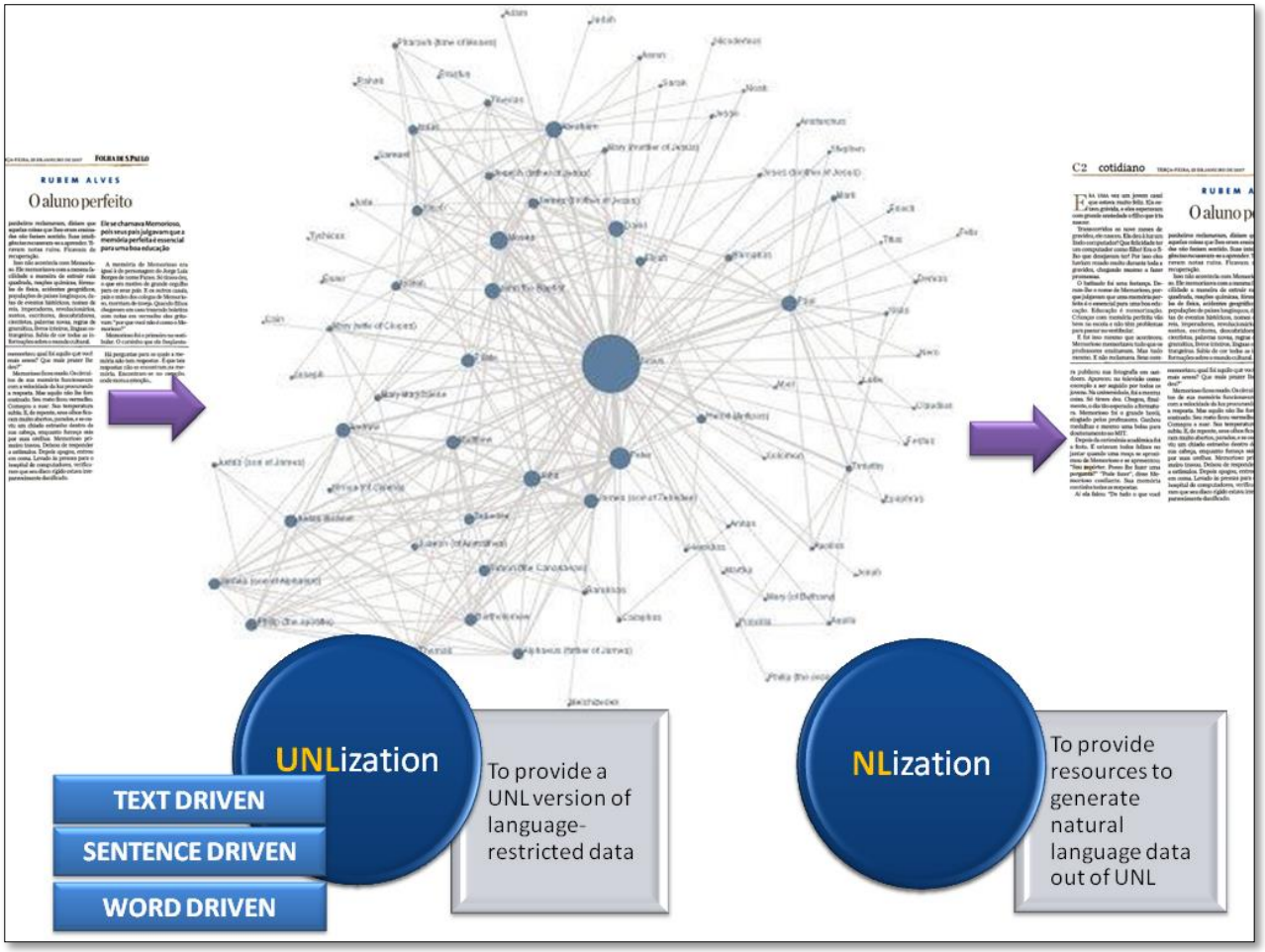


Figure 1.2: UNL System

As shown in Figure 1.2, UNLization can be done on any natural language document that may include a text, word, or a complete sentence. The process of UNLization makes the complete UNL web (in the form of a UNL graph). NLization process takes this UNL graph as an input and transforms UNL to generate natural language again.

Of the other approaches, the UNL approach is the best approach because it captures the semantic information with the help of UWs, attributes, and relations as described in Section 1.6. Moreover, suppose there are *n* numbers of different natural languages. Now, using the approach of UNL for converting those *n* natural languages into each other, *2*n* number of possible translations or mappings need to be done. This is because only two conversions need to be done for that particular natural language, means from that natural language to UNL and then from UNL to that natural language. If this UNL approach is not followed then it can be very well witnessed that *n*(n-1)* is the total number of conversions required for converting every natural language to every other natural language as every language needs to be converted into

the other *n-1* languages [81]. In other words, *n* components for generation, and *n* components for analysis are required in the interlingua approach where *n* refers to the total number of languages in translation system as shown in Figure 1.3 [145].

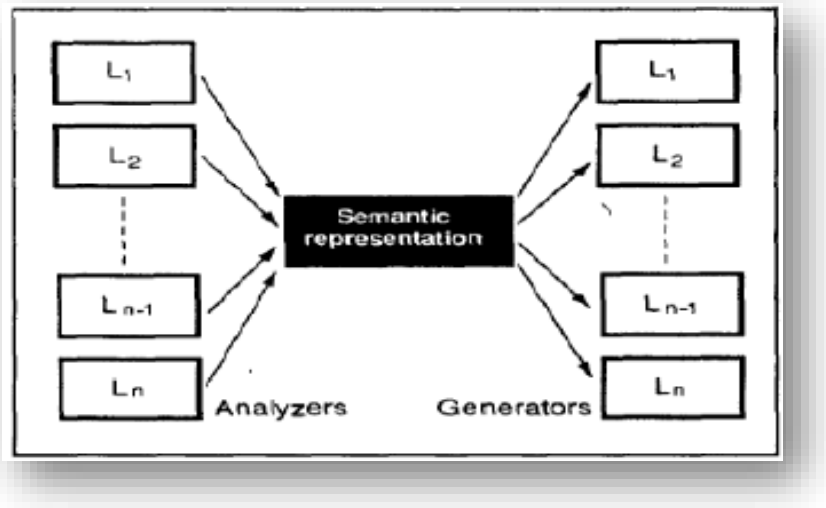


Figure 1.3: Interlingua Translation Between *n* Languages

UNL is not a human language. It is not at all expected that one should speak UNL or should communicate in UNL, rather it is expected that people should use UNL and should communicate through UNL in the same invisible and natural way they do with other declarative and procedural languages which are pervasive in everyday applications. As no one is required to know about HTML to browse the Internet or even to create websites, similarly one would be able to UNLize documents by extracting the information needed without any knowledge of UNL. Therefore, UNL is a formal language designed not for humans, but for computers. To handle natural languages, just like other logical systems, UNL provides the semantic and linguistic infrastructure for computers and not for humans. UNL does not tell about the full meaning of sentences or texts. Instead, it tries to capture the core or consensual meaning. So, UNL's scope and goal do not include poetry, idioms and phrases, feelings, and other indirect communicative behaviors. This is the main reason why UNL has not been exactly an interlingua-based machine translation, even though machine translation is one of the possible as well as more obvious and promising uses of UNL. Currently, the main goal of the UNLization process is to map the information that is verbally elicited in the surface structure of written texts into a language-independent and machine-traceable database. In this sense, the UNL representation has been an interpretation rather than a translation of a given text [13].

This feature of UNL (language-independence and machine traceability) can be exploited for question answering systems. UNL can also be used for other NLP applications like sentiment analysis, text summarization, and machine translation *etc*.

1.5 History of UNL

The UNL Programme started in 1996, as an initiative of the Institute of Advanced Studies of the United Nations University in Tokyo, Japan [161]. For developing and managing UNL Programme, the United Nations University set up an autonomous organization, the UNDL foundation in January 2001. In order to fulfill its mission, it inherited the mandate of implementing the UNL Programme from the UNU/ IAS. Its headquarters is located in Geneva, Switzerland. From the foundation, a group of University departments from all areas of the world has been tied up for developing UNL. That's the UNL society, a global-scale network of research and development teams, involving several specialists in computer science and linguistics, who are at work creating the linguistic resources and developing the web structure of the UNL system [161]. The technological support as well as coordination to implement the programme is provided by the UNDL foundation. The programme has already crossed the major milestones. The overall architecture of the UNL system has been developed with a set of basic software and tools necessary for its functioning [161]. A huge amount of linguistic resources are already under development for the various native languages and have been collected in the last few years. Moreover, the availability of technical infrastructure for the expansion of these resources is also facilitating the participation of many more languages in the UNL system. UNDL foundation has formally defined the specifications of UNL. UNL project aims to create an Interlingua for all major natural languages. It was initially started for the six official languages of the United Nations and other widely spoken languages, namely, Hindi, Arabic, Punjabi, Chinese, English, French, German, Indonesian, Italian, Japanese, Portuguese, Russian, and Spanish. IIT Bombay, a member of the team is responsible for developing UNL modules for Hindi [83]. Active development of UNL modules for Punjabi and Hindi language is also being done at Thapar Institute of Engineering and Technology, Patiala, India [4-9, 81-83, 135-137, 162-164].

1.6 Building Blocks of UNL

UNL represents information sentence by sentence [156]. The UNL system, its

component and first version specifications had been provided by Uchida *et al*. (1999) [158]. In UNL, three different types of semantic units are used to convey the information, namely, Universal Words (UWs), Universal Attributes, and Universal Relations.

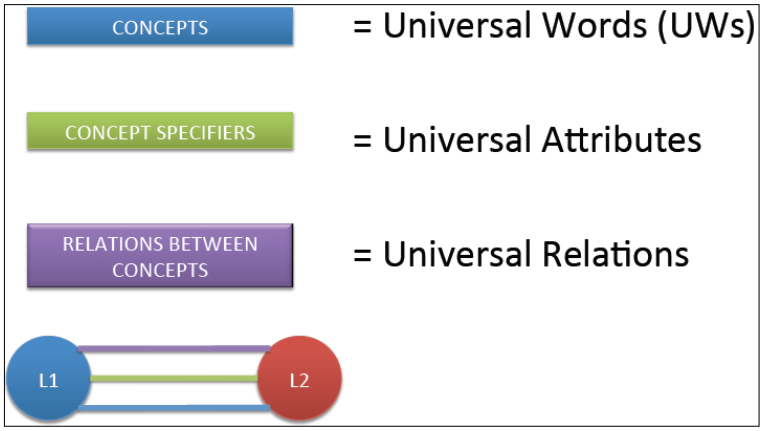


Figure 1.4: Building Blocks of UNL

Each sentence is converted into UNL graph also known as a hyper-graph. UWs represent the concepts and UNL relations specify that what role a word in a sentence will play. The subjective meanings which an author intends are expressed with the help of UNL attributes. The specifications of UNL have been defined formally by UNDL foundation [153].

1.6.1 Universal Words

The nodes in UNL graph represent Universal Words (UWs*)*. These UWs forms the vocabulary of UNL. These words are represented by nodes in a UNL graph. The nodes of the UNL graph are connected by relations and, if required, modified by attributes. The concepts of universal words are divided into four categories, *i.e.*, nominal, verbal, adjective, and adverbial concepts [37]. A universal word is a string of characters followed by a list of constraints. In order to link every universal word of the UNL ontology with the UNL documents, UNL uses the concept of KCIC (Key Concept in Context). To realize this linkage of UWs crossing UNL documents, registration of every universal word in the UNL ontology is must [155]. UWs can be temporary or permanent [161]. UW is described with the syntax as given in (1.1).

<UW>::= <headword> [<constraint list>] …(1.1)

Here, the headword refers to an English word/phrase/sentence, which for a given set of concepts is elucidated as a label. This is also known as a basic Universal Word. In

order to explain the sense of the word, a constraint list is used. An example of Punjabi language dictionary entry is given below.

[ਖ਼ਰਗੋਸ਼]{}"hare"(LEX=N,POS=NOU,NUM=SNG,GEN=MCL);

Here,

"*LEX"* specifies lexical category, "*N"* specifies noun, "*POS"* represents part-of-speech, "*NOU"* represents common noun, "*NUM"* represents number whose value could be either "*SNG"* for singular or "*PLR"* for plural, and "*GEN"* represents gender, "*MCL"* represents masculine.

Punjabi language dictionary has been explained in more detail in section 4.2 of Chapter 4 and section 6.2 of Chapter 6 of this thesis. The objective information of concept in headword is represented by UNL relations that are contained in this list. This constraint list is also used for disambiguation by restricting these different concepts which are represented by an English expression.

1.6.1.1 Temporary UWs

Temporary UWs include those words that represent entities or concepts which are too specific and therefore are not included in the UNL Dictionary (for example "*Vaibhav Agarwal*"); are yet to be lexicalized (for example "*Thaparians*", "*IITians*"); or which cannot be translated ("*3.14159*", "H_2O", "*www.google.com*").

1.6.1.2 Permanent UWs

Permanent UWs are those words which have been included in the UNL Dictionary. Permanent UWs represent concepts that have been already lexicalized. They can be simple, compound or complex. For example "*house*", "*go*", "*hare*", "*tortoise*", "*USA*" *etc*.

1.6.2 Universal Attributes

Universal Attributes are a kind of annotations or comments which are made to hypernodes of a UNL hypergraph or simple nodes. For example "*@past*", "*@interrogative*", "*@def*" *etc*. The syntax of universal attributes is given in (1.2) [161].

<attribute> ::= "@"<attribute name>
<attribute name> ::= <character>+
<character> ::={"a",…,"z" , "_" } …(1.2)

The meaning and description of each symbol are given in Table 1.1.

Table 1.1: Description of Symbols Used to Represent UNL Attributes

Symbol	Meaning
< >	Variable
" "	terminal symbol
::=	is defined as
{ }	disjunction ("or")
+	to be used one or more times
...	to be repeated more than 0 times

A universal attribute is written in lower case words or expressions and they convey three different kinds of details. The subjective information of a sentence is narrated by the attributes. In a given sentence, the narrator's point of view is exhibited by these attributes [157]. To convey the semantic content of a sentence, there are total eighty-seven attributes (that can be increased with the user-defined ones). UNDL foundation has divided these UNL attributes into eight groups. The description of UNL attributes along with their corresponding groups is given in Table 1.2 [153].

Table 1.2: Description of UNL Attributes

Concept	Attributed as
Logicality of UW	@transitive, @symmetric, @identifiable, @disjointed
Time with respect to the speaker	@past, @present, @future
Speaker's view on aspects of event	@begin, @complete, @continue, @custom, @end, @experience, @progress, @repeat, @state,@just, @soon, @yet
Speaker's view of reference to concepts	@generic, @def, @indef, @not, @ordinal
Speaker's emphasis, focus and topic	@contrast, @emphasis, @entry, @qfocus, @theme, @title, @topic
Speaker's attitudes	@affirmative, @confirmation, @exclamation, @humility, @imperative, @interrogative, @invitation, @polite, @request, @respect, @vocative
Speaker's feelings and	Attributes to represent ability: @ability

judgments	Attributes to represent beneficially: @get-benefit, @give-benefit Attributes to represent conclusion: @conclusion, @consequence Attributes to represent condition: @sufficient Attributes to represent consent/dissent: @consent, @dissent, @grant, @grant-not Attributes to represent expectation: @although, @discontented, @expectation, @wish Attributes to represent intention: @insistence, @intention, @want, @will, @need, @obligation, @obligation-not, @should, @unavoidable Attributes to represent possibility: @certain, @inevitable, @may, @possible, @probable, @rare, @unreal Attributes to represent emotion: @admire, @blame, @contempt, @regret, @surprised, @troublesome
Convention description	@passive, @pl, @angle_bracket, @brace, @double_parenthesis, @double_quote, @parenthesis, @single_quote, @square_bracket

In the example sentence, "*The girl throw tomatoes in the bedroom*", attributes are needed to:

a. Denote the head of expression ("*throw*", *i.e.*, "*@entry*").
b. Express plurality in the object ("*tomato*", *i.e.*, "*@pl*")
c. Specify definite reference to place ("*bedroom*", *i.e.*, "*@def*")
d. Specify definite reference to agent ("*girl*", *i.e.*, "*@def*")

Universal attributes are thus used to incorporate information about predictions modality, aspect and time of an event, gender and/or number *etc*.

1.6.3 Universal Relations

Universal relations are also called as "links". These are labeled arcs connecting one

node to another node in a UNL graph. The relations represent the two-place semantic predicates holding between two UWs. The repertoire of relations is defined in the UNL specifications and it is not open to frequent additions. In UNL, relations between UWs represent semantic functions or thematic roles (such as object, agent, instrument, *etc*.) [161]. These semantic functions are binary, universal and are directed (from a source to a target). The relations between the two nodes are represented as $rel\_name(node_1, +att_1; node_2, +att_2)$.

Here, rel_name is the name of the relation, $node_1$ and $node_2$ are the two nodes between which the relations hold attributes; att_1 *and* att_2 are the attributes which are added to $node_1$ and $node_2$ respectively. Here, although there are only two attributes but there can be any number of additional attributes which can be added.

For example, consider UNL representation of English sentence "*The girl throw tomatoes in the bedroom*" given in 1.3.

{unl}
plc(throw(icl>do)@entry, bedroom(icl>facilities)@def)
obj(throw(icl>do)@entry, tomato(icl>food)@pl)
agt(throw(icl>do)@entry, girl(icl>person)@def)
{/unl} …(1.3)

UWs, universal attributes, and universal relations of UNL given in (1.3) are shown with the help of UNL graph in Figure 1.5.

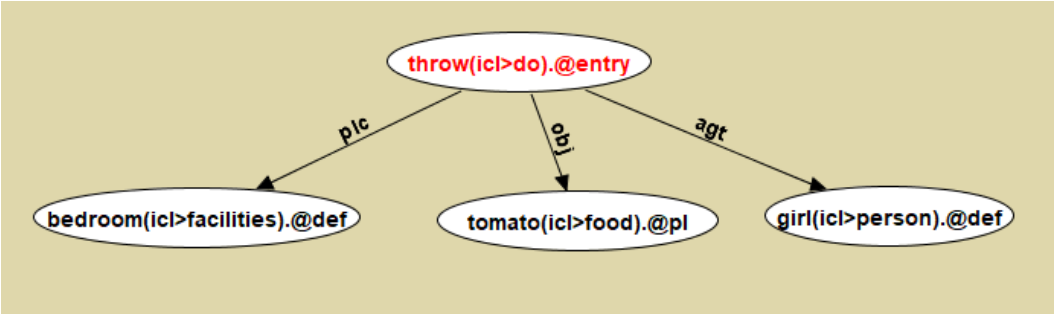


Figure 1.5: UNL Graph of Sentence "*The girl throw tomatoes in the kitchen*"

In the given example, "*agt"* relation specifies relation between agent "*girl*" (who did work), and verb "*throw*". The "*obj*" relation specifies the relation between object "*tomato*", and verb "*throw*". The "*plc*" relation specifies the relation between verb "*throw*", and place "*bedroom*" where the action took place. Finally, attributes represent the circumstances under which the node is used. These are the annotations made to nodes. In the given example sentence, "*@pl*" signifies that UW is used as a

plural. The attribute "*@def*" is a kind of specifying attribute used in case of general specification (normally conveyed by determiners) which is represented by node "*the*". The "*@entry*" attribute represents sentence head. The attributes give additional information that is not expressed via UW or Universal Relations.

For UNL, total forty-six relations have been defined by UNDL foundation [153]. The description of these relations is given in Table 1.3 [159].

Table 1.3: UNL Relations Description

UNL relation	**Description**	**Constituent elements**	**Examples**
1. agt	defines a thing that initiates an action	Agent, Unergative verbs (intransitive verb semantically have an agent subject) like 'sleep', 'snore', 'cough', 'run', etc.	John slept … agt(slept, John) John killed Mary. agt(killed, John) … arrival of John … agt(arrival, John) …play by Shakespeare agt(play, Shakespeare)
2. and	defines a partner to have conjunctive relation	Conjunction	… easily and quickly … and(quickly, easily) … singing and dancing … and(dance(agt>person),sing(agt >person))
3. aoj	defines a thing that is in a state or has an attribute	Stative verbs like 'believe', 'understand', 'know', 'have', 'possess', 'dislike', 'love', 'like', 'contain',	John believes in Mary. aoj(believes, John) John knows Mary. aoj(knows, John) John loves Mary. aoj(loves, John)

		‘include’, ‘involve’, etc.	
		aoj (general attribute)	John is sad. aoj(sad, John) John looks sad …. aoj(sad, John)
4. bas	defines a thing used as the basis (standard) of comparison	Basis	... more than seven. bas(more(aoj>thing,bas> thing), 7) … more than Jack. bas(more(icl>how,bas>thing), Jack(iof>person))
5. ben	defines an indirectly related beneficiary or victim of an event or state	Beneficiary	To give one’s life for one’s country. ben(give(agt>thing,gol>thing,obj>thing), country(icl>region)) It is good for John to … ben(good(aoj>thing),John(iof>person))
6. cag	Defines a thing not in focus that initiates an implicit event that is done in parallel	Co-agent	… to walk with John. cag(walk(agt>volitional thing), John(iof>person)) To live with … aunt. cag(live(agt>volitional thing),aunt(icl>person))
7. cao	defines a thing not in focus that is in a parallel state	Co-thing with attribute	... be with you … cao(exist(aoj>thing), you)

8. cnt	defines the content of a concept	Content	The Internet: an amalgamation … cnt(Internet(icl> communication network),amalgamation(icl >harmony)) A language generator "deconverter"… cnt(language generator, deconverter.@double_quote)
9. cob	defines a thing that is directly affected by an implicit event done in parallel or an implicit state in parallel	Affected co-thing	… dead with Mary. cob(die(obj>living thing), Mary(iof>person))
10. con	defines a non-focused event or state that conditions a focused event or state	Condition	If you are tired, we will go straight home. aoj:01(tired(aoj>thing), you) con(go(icl>move(agt>thing,gol> place,src>place)), :01)
11. coo	defines a co-occurrent event or state for a focused event or state	While	… worked while … talked coo(worked, talked)
12. dur	defines a period of time during which an event	During	John worked during … dur(worked, meeting)

	occurs or a state exists		
13. equ	defines an equivalent concept	Equivalent	The deconverter (a language generator) … equ(deconverter, language generator.@parenthesis)
14. fmt	defines a range between two things	Range/from-to	… from a to z. fmt(z(icl>letter), a(icl>letter)) … from Osaka to New York. fmt(New York(iof>city),Osaka (iof>city))
15. frm	defines an initial state of a thing or a thing initially associated with the focused thing	Origin	A visitor from Japan … frm(visitor(icl>person),Japan (iof>country))
16. gol	defines a final state of object or a thing finally associated with the object of an event	Final state of verbs of change like ‘give’, ‘send’, etc.	… gave … to Mary. gol(gave, Mary) … sent … to Mary. gol(sent, Mary)
17. icl	defines an upper concept or a more general concept	Included/a kind of	A bird is a (kind of) animal. icl(bird(icl>animal),animal(icl >living thing))
18. ins	defines an	Instrument	… look at stars through a

	instrument to carry out an event		telescope. ins(look(agt>thing,obj>thing), telescope(icl>optical instrument)) … write with a pencil. ins(write(agt>thing,obj>thing), pencil(icl>stationery))
19. int	defines all common instances to have with a partner concept	Intersection	…. an intersection of tableware and cookware … int(tableware(icl>tool),cookware (icl>tool))
20. iof	defines a class concept that an instance belongs to	An instance of	Tokyo is a city in Japan. iof(Tokyo(iof>city), city in Japan)
21. man	defines a way to carry out an event or the characteristics of a state	Manner	… move quickly. man(move(agt<thing,gol>place ,src>place), quickly) … often visit ... man(visit(agt>thing, obj>thing), often)
22. met	defines a means to carry out an event	Method/means	… solve … with dynamics. met(solve(icl>resolve(agt>thing, obj>thing)),dynamics(icl>science)) … separate … by cutting … met(separate(agt>thing,obj>thing, src>thing),cut(agt>thing,obj> thing,opl>thing))

23. mod	defines a thing that restricts a focused thing	Modification	The whole story … mod(story(icl>tale),whole(mod <thing)) A master plan …. mod(plan(icl>idea),master(mod <thing))
24. nam	defines a name of a thing	Name	... son “Hikari” nam(son(icl>relative), Hikari)
25. obj	defines a thing in focus that is directly affected by an event or state	Unaccusative verbs like ‘die’, ‘fall’, ‘melt’, etc.	John died. obj(died, John) The snow melts. obj(melts,snow)
		obj (direct object)	John killed Mary. obj(killed, Mary) John knows Mary. obj(knows, Mary) John loves Mary. obj(loves, Mary)
		obj (indirect object of monotransitive verbs like ‘depend’, ‘believe’, ‘laugh’, etc.)	… depends on Mary. obj(depends, Mary) … believes in Mary. obj(believes, Mary)
		Object	… construction of the building ... obj(construction, building) … interest in Physics …

			obj(interest, Physics) … visit to London. obj(visit, London)
26. opl	defines a place in focus affected by an event	Affected place	… pat … on shoulder …. opl(pat(icl>touch(agt>thing,obj >thing,opl>thing)),shoulder(pof >trunk)) … cut … in middle … opl(cut(agt>thing,obj>thing,opl >thing), middle(icl>place))
27. or	defines a partner to have disjunctive relation to	Disjunction	… stay or leave … or(leave(agt>thing,obj>place),stay (icl>remain(agt>thing))) Is it red or blue? or(blue(icl>color), red(icl>color))
28. per	defines a basis or unit of proportion, rate or distribution	Proportion/rate/di stribution	… hours a day. per(hour(icl>period),day(icl> period))
29. plc	defines a place where an event occurs, or a state is true, or a thing exists	Place	Made in India. plc(made, India)
30. plf	The place where an event begins or a state becomes true	Initial place of verbs of motion like ‘go’, ‘travel’, ‘walk’, ‘come’, etc.	John came from NY. plf(came, NY)

31. plt	defines a place where an event ends or a state that becomes false	Final place	… to travel to Patiala. plt(travel(agt>volitional thing), Patiala(iof>city))
32. pof	indicate a concept of which a focused thing is a part	Part of	The preamble of a document … pof(preamble(icl>information), document(icl>information)) … the initials of Machine Translation … pof(initial(icl>letter), machine translation)
33. pos	defines the possessor of a thing	Possessor	John's dog. pos(dog(icl>aminal),John(iof >person)) My book. pos(book(icl>document), I)
34. ptn	defines an indispensable non-focused initiator of an action	Partner	… compete with John … ptn(compete(agt>thing,ptn>thing) , John(iof>person)) … share … with the poor. ptn(share(icl>divide(agt>thing,obj >thing)),poor(icl>person))
35. pur	defines the purpose or objective of an agent of an event or the purpose of a thing that	Purpose	John works for money. agt(work(icl>do),John(iof>person)) pur(work(icl>do), money) … budget for research … pur(budget(icl>expense),research

	exists		(icl>study))
36. qua	defines the quantity of a thing or unit	Quantity	Two cups of coffee … qua(cup(icl>tableware),2)) qua(coffee(icl>beverage), cup(icl>tableware)) … many kilograms … qua(kilogram(icl>unit),many(qua <thing)) … two dogs … qua(dog(icl>animal), 2)
37. rsn	defines a reason why an event or a state happens	Reason	He goes … because of … illness. rsn(go(icl>do), illness(icl>thing)) … known for … beauty. rsn(known(aoj>thing),beauty(icl >abstract thing))
38. scn	defines a scene where an event occurs, or state is true, or a thing exists	Scene	… appear on a program. scn(appear(icl>occur),program(icl >thing))
39. seq	defines a prior event or state of a focused event or state	After	John worked and left … seq(left, worked)
40. shd	defines a number, a mark or a thing that shows the position of a sentence, a	Sentence head	Chapter 2 Relation shd(relation(icl>sate),chapter(pof >book)) mod(chapter(pof>book), 2)

	paragraph or a chapter in a document or a book		
41. src	defines the initial state of an object or thing initially associated with the object of an event	Initial state of verbs of change like ‘take’, ‘retrieve’, etc.	… changed from red … src(change(icl>occur),red(aoj >thing))
42. tim	defines the time an event occurs or a state is true	When	… came yesterday … tim(came, yesterday)
43. tmf	defines the time an event starts or a state becomes true	Since when	… worked since early … tmf(worked, early)
44. tmt	defines the time an event ends or a state becomes false	Until when	… worked until late … tmt(worked, late)
45. to	defines a final state of a thing or a final thing (destination) associated with the focused thing	Destination	… train for London … to(train(icl>thing),London(icl >city))
46. via	defines an intermediate	An intermediate place or state	… goes to … via New York. via(go(icl>do),New York(icl

	place or state of an event		>place))

The representation of UNL should be semantically saturated and deictics are expected to be replaced in UNLization. Therefore, natural language pro-forms (like *"it", "he", "they", "she" etc*.) should be substituted by their corresponding antecedents. In cases when the substitute is not available, these are represented by the null UW *"00"* combined with universal attributes, if required. Such main cases are explained as follows.

- **Exophora** like *"you", "I", "we" etc*., are represented by the null UW *"00"* followed by the person attributes (*"@1"*, for first person singular; *"@2"*, for second person singular; *"@3"*, for third person singular; *"@1.@pl"*, for first person plural; *"@2.@pl"*, for second person plural; and *"@3.@pl"*, for third person plural).
- **Indefinite pronouns** like *"anyone", "none", "everything" etc*., are represented by the null UW *"00"* followed by determiner attributes (*"anyone" = "00.@any.@person", "none" = "00.@no", "everything" = "00.@every.@thing" etc*.).
- **Interrogative pronouns** like*"whom", "who", "where" etc*., are represented by the null UW *"00"* followed by the attribute *"@wh"* (*"who" = "00.@wh", "whom" = "00.@wh", "where" = "00.@wh" etc*.). The difference between them is determined by the relation in whom they appear. For example, in case of *"agt"* relation if target argument is *"00.@wh"*, then it is to be understood as *"who"*; in case of *"tim"* relation if target argument is *"00.@wh"*, then it is to be understood as *"when"*; in case of *"plc"* relation if target argument is *"00.@wh"*, then it is to be understood as *"where"*; and so on.

Since UNL captures information using UWs, universal attributes, and universal relations, therefore it can be said that UNL based question answering systems would be syntactically very efficient to handle all the complex questions.

1.7 Working Principle of UNL for QA System

UNL captures the representation of the semantics of a natural language text. This feature of UNL can be exploited in other NLP tasks such as question answering,

sentiment analysis, machine translation, and text summarization *etc.* Using any existing programming/scripting language the knowledge extraction becomes very easy. Consider an example sentence given in (1.4).

"The book on the table about Paris without pictures." …(1.4)

The UNL graph of this example sentence in shown in Figure 1.6.

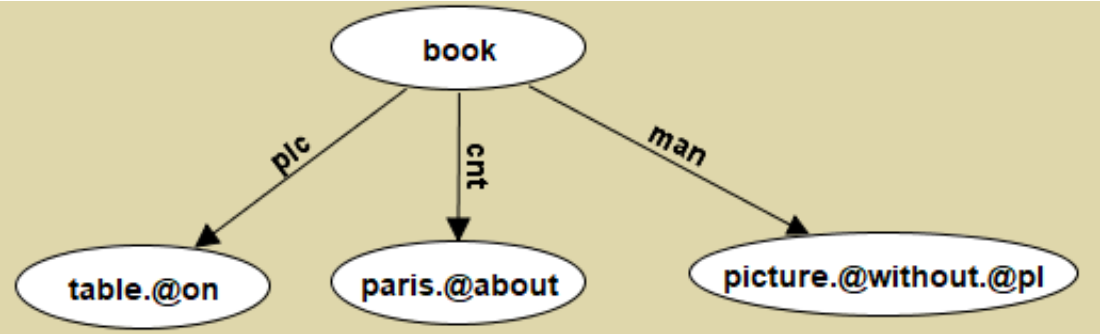


Figure 1.6: UNL Graph of (1.4)

Now, let's say user asks the question "*Where is book*"?

Having identified the main relation as *"plc", i.e.*, place, the node highlighted in Figure 1.7 provides the desired answer which is *"table"*.

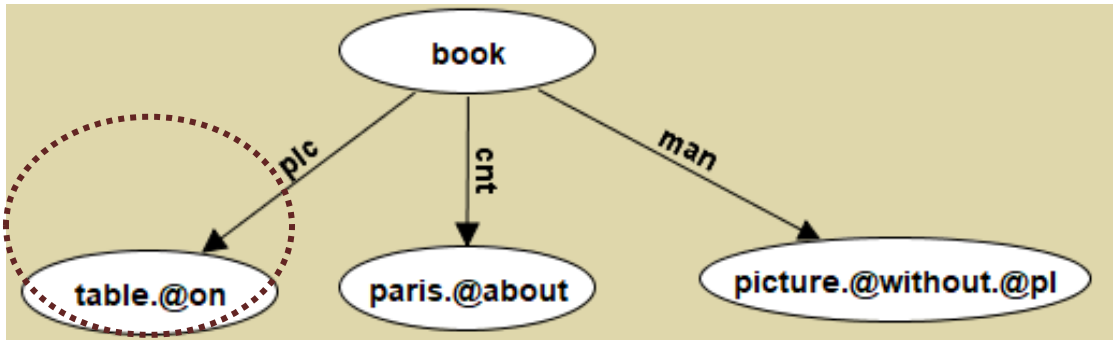


Figure 1.7: Highlighted Entry Showing Answer Node in UNL Graph of (1.4) for the Asked Question "*Where is book*"

Consider another complex sentence given in (1.5).

CSE department was awarded the innovation prize in 2016 by IIT Mumbai for a new type of online question answering system. …(1.5)

Figure 1.8 illustrates the UNL graph of this example sentence.

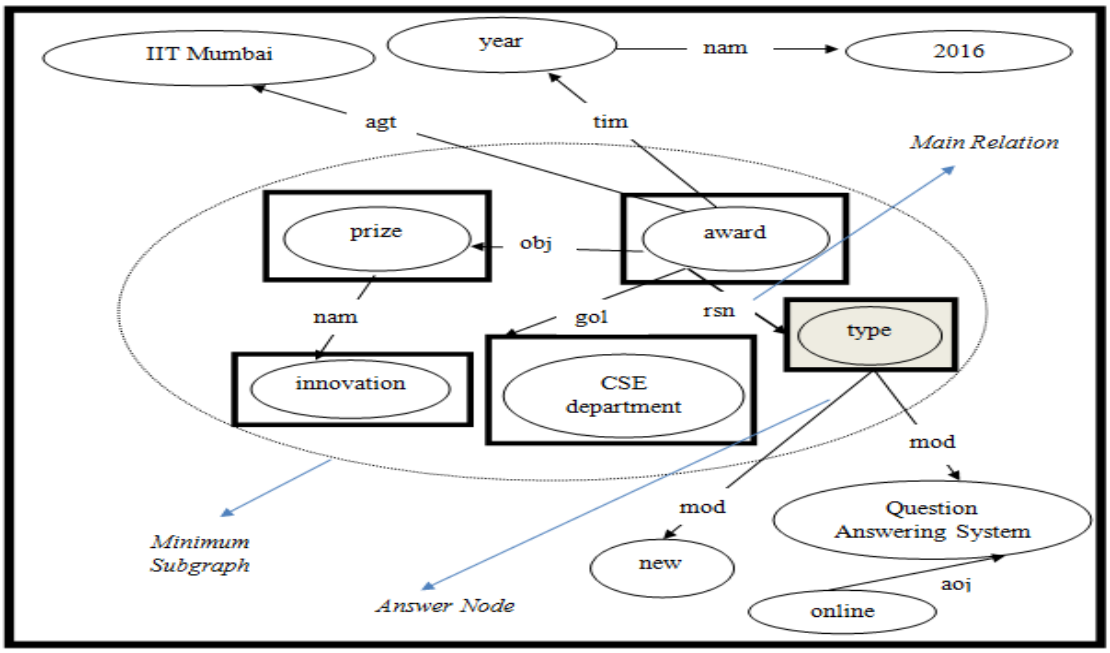


Figure 1.8: UNL Graph of Example Sentence (1.5)

Now, let's say user asks the question "*Why was CSE department awarded the innovation prize*"?

Having identified the main relation as *"rsn" i.e.*, reason, the subgraph given in Figure 1.9 provides the desired answer which is "*a new type of online question answering system*".

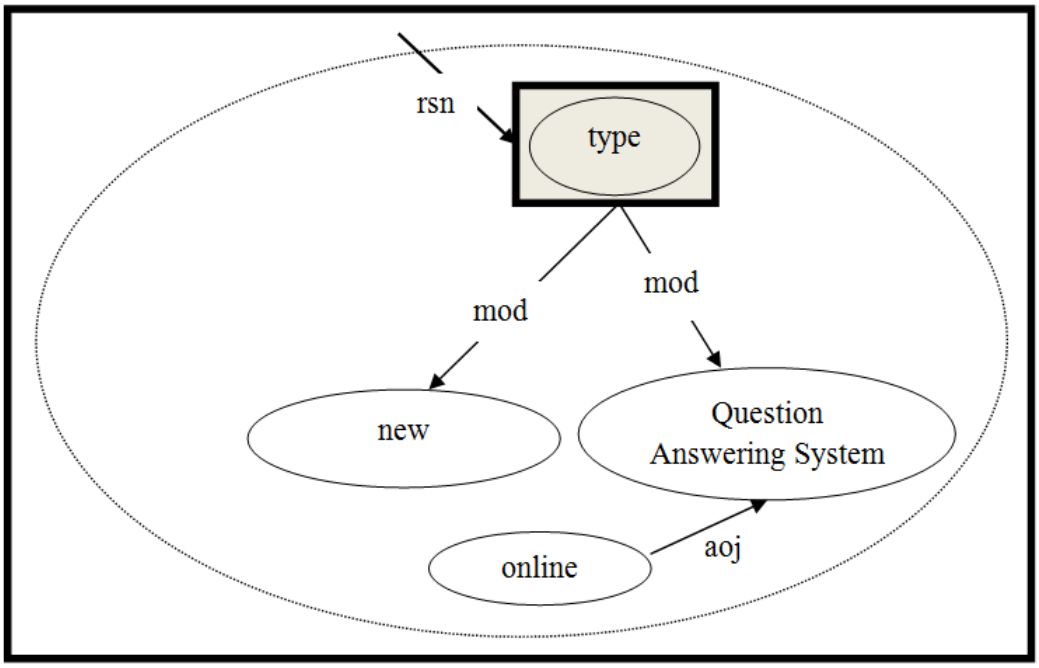


Figure 1.9: UNL Graph of the Answer to Question "*Why was CSE department awarded the innovation prize*?"

1.8 Gap Analysis

Based on the study of various QA systems, some of the research gaps that have been identified are as follows.

- *Non Availability of QA system for the Punjabi language*

  The Punjabi language is currently considered as world's 10^{th} most widely spoken language [53]. In the field of development and computerization, there are relatively fewer efforts for this language. At present, there is no QA system available, which provides scalability to support multilingualism and Punjabi language. Therefore, there is a need to design and develop a question answering system for the Punjabi language. This system will certainly be very helpful for more than 91 million Punjabi language users [118].

- *Lack of integration with other QA system*

  IAN (Interactive ANalyzer) and EUGENE (dEep-to-sUrface GENErator) are online tools developed by UNDL foundation and released in 2012 for UNLization and NLization processes respectively. The earlier tools became obsolete because they were very complex and it was difficult to integrate them on a central site. IAN and EUGENE have been accepted globally as a standardized tool for UNLization and NLization respectively. Globally more than 40 languages are using these tools which have taken participation in UNL programme. The proposed UNL based QA system will use IAN and EUGENE.

- *Lack of support for multilingualism*

  None of the existing QA systems provide scalability to support multilingualism. Although some present systems do support multiple languages but they cannot be extended to support other foreign languages without changing their underlying architecture and implementation. The development of a UNL based QA system for the Punjabi language would support all the languages which are part of the UNL programme or going to be its part in the future. This system will definitely be a unique and useful system of its own kind as it would not require to change the architecture and implementation to support other foreign languages.

- *Lack of integration of different NLP applications*

  Existing NLP applications across the globe cannot integrate and leverage for integrating other NLP based applications. The platform for the proposed UNL

based QA system can be used for developing other UNL based NLP applications like sentiment analysis, machine translation, text summarization *etc*.

- *Complexity*

  Suppose there are *n* numbers of different natural languages. Now using the approach of UNL for converting those *n* natural languages into each other, *2*n* number of possible translations or mappings that need to be done. This is because now only *2* conversions need to be done for that particular natural language *i.e.,* from that natural language to UNL and then from UNL to that natural language. If this approach is not followed, the total number of conversions to convert every natural language to every other natural language would have been *n*(n-1)* as every language needs to be converted into the other *n-1* languages.

1.9 Objectives

The core objective of this study is to develop a UNL based question answering system for Punjabi language. To accomplish this task following objectives were proposed.

i. To study and analyze various QA systems and their approaches.
ii. To develop and test analysis module that would convert Punjabi to UNL with the help of IAN.
iii. To develop and test UNL crawler for searching UNL repository for corresponding input Punjabi question.
iv. To develop and test Optimizer for optimizing and inferring the search results.
v. To develop and test Generation module for the Punjabi language with the help of EUGENE to generate answers from the UNL corresponding to the answer.

1.10 Contributions to Thesis

The major outcomes of this research work are as follows.

i. Study and comparison of existing QASs has been done on the basis of important parameters like corpus, evaluation metrics used, evaluation metrics values, and domain *etc.*
ii. Various gaps in the existing QASs has been identified on the basis of important parameters like, availability of source code, scalability, online availability, multilingual support, is working algorithms of all components

explained, and can question answering system integrate other NLP applications *etc.*

iii. A web based language independent architecture for QAS has been proposed to address the identified gaps.

iv. A web based interface for the QAS has been developed on top of the proposed architecture. UNL crawler and optimizer has been developed and integrated in this interface.

v. UNLization/ IAN module for Punjabi language has been developed.

vi. NLization/ EUGENE module for Punjabi language has been developed.

vii. Analysis and generation dictionaries for Punjabi language have been created.

viii. The developed QAS has social significance. It has been tested on UNL-EOLSS and Agro-Explorer corpora which are based on water resources and farming.

ix. Developed 'Share' functionality by which worldwide users can share their UNLization/ IAN and NLization/ EUGENE resources so as to make their natural language available for use (in the developed QAS).

x. Implemented OTP based authentication feature for 'Share' functionality.

Chapter Summary

In this chapter, the introduction to question answering systems, their classification, and their components have been documented. Since the proposed question answering system is UNL based, therefore an introduction to UNL and its building blocks, *i.e.*, Universal Words, Universal Attributes, and Universal Relations have been described in detail in this chapter. This chapter highlights the importance and use of UNL in NLP applications. This chapter explains the working principle of UNL based question answering system with the help of example sentences. This chapter also highlights the gap analysis and the objectives that were framed for this thesis. The major contributions of this research work have been documented in this chapter.

Chapter 2

Background Theory and Literature Review

During the last couple of years, in the field of Natural Language Processing (NLP), QA systems have been an area which had attracted researchers. Currently NLP and machine learning techniques have been used in almost every domain like biomedical (Arvind and Rajeev, 2014), academics (Sujata and Parteek, 2017), network security (Sommer and Paxson, 2010), and agriculture (Surve *et al.,* 2004) [15, 146, 140, 147] *etc*. An accurate and language independent QA system can prove very useful in it. This chapter focuses on the analysis and study done of various QA systems, and comparison of some important QA systems. After doing an exhaustive survey on various QA systems, it has been observed that the proposed UNL based QA system will definitely be a major step in natural language processing and removing the language barrier. This chapter also contains the study of various UNL based activities. In this chapter, the literature survey performed has been documented and divided into two parts, *i.e.*, research activities in QA systems and research activities in UNL.

2.1 Research Activities in QA Systems

Historically, the best-known early QAS was BASEBALL, a system developed by Green *et al*. (1963) in Lincoln laboratory (MIT) to answer the questions about baseball games played in the American league over one season [55]. It was a system that answers to the questions that were phrased in ordinary English on the stored data. The question was read by the program from punched cards. After the words and idioms were looked up in a dictionary, the phrase structure and other syntactic facts were determined for content analysis, which listed attribute-value pairs specifying the information given and the information requested. From the data matching the specifications, the requested information was then extracted, and necessary processing was pursued and then the answer was displayed. This system was limited in the context of baseball data. The data was the place, month, day, scores and teams for each game in the American league for one year. In that limited context, the data were simple, a small vocabulary was sufficient, and the subject-matter was familiar.

The most well-remembered other early work in this field is the LUNAR system, which was designed in 1971 as a result of the Apollo moon mission [171]. The system

was developed to enable lunar geologists to conveniently observe, match and calculate the chemical analysis data on lunar soil and rock composition that was accumulating. The knowledge that the current system contained about the use of English and the corresponding meanings of words and phrases was related to those English constructions which were relevant to the system's database of chemical analysis data, which had a very limited and simple structure. For computing the answer, the user's request was converted into a custom-made program. This was done by the module which worked as an automatic programmer. The system knew how geologists normally referred to the minerals, measurements, and elements contained in its database. The database table layout details were also known to the module, and the module also knew the correspondence between the two.

Lehnert (1977) proposed QUALM, a computer program that used a conceptual taxonomy of thirteen conceptual classes [88]. The processes in QUALM were divided into four phases, *i.e.*, conceptual categorization, inferential analysis, content specification, and retrieval heuristic. QUALM provided a concrete criterion for judging the strengths and weaknesses of store representations. As a theoretical model, QUALM was intended to describe general question answering, where question answering was viewed as a verbal communication device between people.

Bronnenberg *et al*. (1980) developed a PHLIQAl system at Philips research laboratories [28]. This system answered short questions against a database consisting of fictitious data about computer installations in Europe. The questions were first translated into a formal language which was then used to access the database. The translation process was divided into three stages, *i.e*., English-oriented Formal Language (EFL), World Model Language (WML), and Data Base Language (DBL).

Ask Jeeves nowadays known as ask.com was a question answering focused web search engine founded in 1995 by Garrett Gruener and David Warthen in Berkeley, California [16]. Ask launched a question answer community to generate answers from people (as opposed to search algorithms) then combined this with its question-and-answer repository, utilizing its extensive history of archived query data to search sites that provided answers to questions which people had asked. In the case, when no answers were available from its own resources, the company outsourced to third-party search to avoid the situation which provided the comprehensive web search matches that it had gathered itself.

Moldovan *et al*. (1999) proposed a tool for surfing the answer net known as LASSO [104]. The architecture of LASSO comprises of three modules, *i.e.*, question processing module, paragraph indexing module and answer processing module. They proposed a hierarchical taxonomy that classified the question types into nine classes, each of which was divided into a number of subclasses. These question classes and subclasses covered all 200 questions in the corpus of TREC-8 (Eighth Text Retrieval Conference). Their research relied on NLP techniques in novel ways to find answers in large collections of documents. The question was processed by combining syntactic information with semantic information that characterized the question, for example, question type or question focus, in which eight heuristic rules were defined to extract the keywords used for identifying the answer. The research also introduced paragraph indexing where retrieved documents were first filtered into paragraphs and then ordered.

Prager *et al*. (1999) had shown the use of predictive annotation for question answering in TREC-8 [115]. The idea of their approach was just to index the answers. For this about 20 classes of objects had been established that can be identified in the text by shallow parsing, and by indexing and annotating the text with these labels, which were called QA-Tokens. The class was identified for a given question and in order to include the appropriate tokens, the question was modified accordingly. In order to rank and return short passages of text rather than documents, the search engine was modified. In later stages of analysis, the QA-Tokens were used in order to extract the supposed answers from these returned passages. At last, all the potential answers were ranked with the help of a novel formula, according to which it was determined that which ones among them were most likely to be correct.

Harabagiu *et al*. (2000) proposed an open domain question answering system called FALCON [60]. The newly proposed system, FALCON, was characterized by additional features and components. They generated a retrieval model for boosting knowledge in the answer engine by using WordNet for semantic processing of questions. For short answers, a score of 58% was generated by FALCON in TREC-9. Also, in order to overcome the main limitation that appeared in LASSO, they provided a justification option to rule-out erroneous answers to provide only one answer.

Cooper and Ruger (2000) developed a question answering system that uses the CMU link parser (Sleator and Temperley, 1991), Princeton University's WordNet (Miller,

1995), the REX system for XML Parsing (Cameron, 1998) and the managing gigabyte search engine (Witten *et al*., 1999) [38, 138, 98, 31, 170]. During the paragraph retrieval by managing gigabyte search engine, at the document paragraph lookup stage, the information of answer type and possible keywords were used to extract documents from the text database that might contain answers to the question. The proposed system compared favourably against other systems competing in TREC-9. The emperical result of the proposed system was not mentioned and no failure analysis was done.

Gaizauskas and Humphreys (2000) described an approach to question answering that was based on linking an IR system with an NLP system that performed reasonably thorough linguistic analysis [51]. The question was treated as a query by the IR system and a set of top-ranked passages and documents was returned. These set of returned passages and documents was analyzed by the NLP system and question was also parsed.

Hermjakob (2001) used machine learning based parsing and question classification for question answering [64]. He empirically demonstrated that when a Penn Treebank training corpus which is enriched semantically was augmented with an additional question treebank, the question parsing improved. He also demonstrated that parse trees have to be semantically richer and structurally more oriented towards semantics than other treebanks. An overall Mean Reciprocal Rank (MRR) of the system was 0.318.

Li and Roth (2002) proposed a two-layered taxonomy which had six super (coarse) classes and fifty fine classes [90]. Questions were classified based on that taxonomy using two main approaches, *i.e.*, rule-based classifiers and machine learning classifiers. Apparently, the rule-based classifier was a straightforward way to classify a question according to a taxonomy using a set of predefined heuristic rules. In the machine learning approach, a machine learning model was designed and trained on an annotated corpus composed of questions which are labeled.

A method for learning surface text patterns automatically had been developed by Ravichandran and Hovy (2002) [122]. The utility of these surface text patterns for open domain question answering systems had been explored by them. A tagged corpus from the world wide web had been built by providing few manually created examples

of every type of question to Altavista. From these returned documents, patterns are automatically extracted and standardized. The answers to any new question were then found by these patterns.

Radev *et al*. (2002) proposed a QA system called NSIR, pronounced *"answer"*, which used a flat taxonomy with seventeen classes [120]. These were person, number, place, date, definition, organization, description, abbreviation, known for, rate, length, money, reason, duration, purpose, nominal, other. NSIR takes between 5 and 30 seconds per question depending on the number of documents to be downloaded from the web and on the number of phrases to extract. The current version of NSIR did not include query modulation that was the process of converting a question to the best query for a given search engine.

Xu *et al*. (2003) proposed a question answering system in which text patterns were automatically acquired in order to enhance the poor recall of the manual hand-crafting patterns [175]. The proposed system worked on definitional questions and processed them in six different steps. First step deal with question type classification. As a part of second step, documents were pulled from TREC-12 corpus about the question target. In the third step, application of heuristics to the sentences in the retrieved documents was done. Redundancy detection and minimization of irrelevant materials were done in the fourth step to create kernel facts. In the fifth step, ranking of kernel facts was done. In the last step, *i.e.*, step six, duplicate kernel facts were detected after application of some heuristics.

Shih *et al.* (2005) proposed an Academia Sinica Question-Answering System (ASQA) [131]. The proposed system was able to give exact answers to factoid questions of six types. The architecture of their proposed system had four main components, *i.e*., question processing, passage retrieval, answer extraction, and answer ranking. Instead of exploiting surface text information using patterns they employed the named-entity approach to find an answer. The accuracy of their system was only 37.5% for correct answers whereas it was 44.5% for unsupported+correct answers.

Stoyanchev *et al*. (2008) presented a document retrieval experiment on a QAS and evaluated the use of named entities, verb, noun, and prepositional phrases as exact match phrases in a document retrieval query [143]. Their experiment was conducted on WEB dataset which resulted in an enhancement over baseline on various measures

of sentence retrieval and documents. By using automatically generated phrases, a relative enhancement of 20% in mean reciprocal rank for extraction of sentences was achieved. Additionally, a relative improvement of 9.5% was achieved by using phrases which were manually annotated.

Kangavari *et al*. (2008) presented a simple approach to improve the accuracy of a QAS using a knowledge database to directly return the same answer for a question that was previously submitted to the QA system, and whose answer has been previously validated by the user [77]. Question processing and answer processing modules had been improved in their proposed model. The co-occurrence tendency of the text corpus or documents was mined for relation prediction between question and answer candidates on which answer validation relied. The testing had been done on a total of 50 questions and their system showed 92% of the improvement.

Manuel and Riofrio (2010) proposed the architecture of a question answering system for a specific repository of documents but that architecture was not statistically tested, verified or implemented by them [92]. The entire architecture was divided into three modules. When the user entered a question its natural language then its representation and analysis operations were performed by the first module. Searching for documents was a part of the second module. The selection of the response was done by the third module.

Shizhu *et al*. (2014) developed CASIA@V2 which was an MLN-based (Markov Logic Network) question answering system over linked data [132]. Their system converted questions in natural language into structured queries automatically. The ambiguities of QALD (Question Answering over Linked Data) were removed in three steps. They achieved an F-measure of 36%, average precision of 32%, and an average recall of 40% on QUALD-4 dataset's 50 questions.

Heba *et al*. (2014) developed Al-Bayan which was an Arabic QA system for the Holy Quran [62]. The system only worked on Arabic questions and that too about the holy Quran. The relevant Quran verses were then retrieved and followed by the process of extraction of passage in which the answer from the Quran was contained. They evaluated their proposed system on a manually collected dataset and got an accuracy of 85% by using the top-3 results.

Kotaro *et al*. (2014) developed Forst which was a QA system using basic element at

at NTCIR-11 QA-Lab ytask [79]. Their system had two modules, *i.e.*, for each question format one dedicated module and a second module which had common modules which were called by the first dedicated modules as and when necessary. Short-essay questions which were in the secondary examinations were tackled by their system. In centre test tasks, out of 97, their system achieved 46 points in phase 1, while it achieved 49 points out of 100 in phase 2.

Junwei *et al*. (2014) proposed a Knowledge-Based QA as machine translation [76]. They translated questions to answers based on CYK (Cocke–Younger–Kasami) parsing. The answer as translations of the span covered by each CYK cell was obtained by a question translation method, which first generated formal triple queries as MRs (Meaning Representations) for the span based on question patterns and relation expressions, and then retrieved answers from a given knowledge-base based on triple queries generated.

Di *et al*. (2014) developed the CMU multiple-choice question answering system [44]. This system answers multiple choice English questions for the world history entrance exam. The questions were preceded by short descriptions providing a historical context. For every answer choice, assertions which could be verified were generated by them if the context and instructions specific to questions were given. On the training set, their system achieved an accuracy of 51.6%.

Pragisha and Reghuraj (2014) have developed a natural language QA system in Malayalam using domain dependent document collection as a repository [116]. The user asked questions in Malayalam natural language and a collection of Malayalam documents were analyzed for answer extraction. The proposed system was able to extract answers for factual questions like which, where, and what.

Sean *et al*. (2014) proposed a QA system called as Watsonsim using the Indri, Lucene, Bing and Google search engines, Apache UIMA, Open NLP, and Weka [128]. For analysis of questions in the proposed system, there were two pipelines. General trivia and factoid questions were analyzed in the first pipeline whereas the second pipeline targeted only fill in the blanks type questions.

Malinowski and Fritz (2014) first commenced the research work on real-world images for question answering by proposing a multi-world approach [91]. In a Bayesian network, the uncertainty about the perceived world was represented by this approach.

The high complexity human questions could be handled by this approach and it was able to answer over a wide range of answers like object classes, lists, instances, and counts. They used a combination of visual perception and symbolic reasoning.

Baudis (2015) developed a modular QA system pipeline called YodaQA [18]. YodaQA reunited and boosted research efforts in question answering, providing a modular, open source pipeline for this task. It allows integration of various knowledge-base paradigms, answer production and analysis strategies and using a machine-learned model to rank the answers. A baseline question answering system which was inspired by deep question answering had been supplied with this proposed pipeline.

Hartawan and Suhartono (2015) proposed the use of vector space model in QAS [61]. The objective was to represent knowledge and retrieve the answer for a given question by utilizing vector space model. On the basis of similarity, comparison of the query was done with the knowledge-base. An F-measure of 0.58, recall of 0.662, and precision of 0.548 were achieved. One major limitation of the proposed system was that its average time to answer the questions asked by the users was very high.

Wang and Nyberg (2015) proposed a Long Short-Term Memory Model (LSTMM) for answer sentence selection in QA [168]. To output the relevancy scores of question and answer sentences, words were read sequentially by using stacked BLSTM (Bidirectional Long-Short Term Memory) network. They evaluated their system on the basis of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). They achieved a MAP of 0.7134 and MRR of 0.7913.

A QAS using learning knowledge graphs through conversational dialog had been proposed by Ben *et al.* (2015) [20]. The proposed system answered questions using a knowledge graph. This knowledge graph stored new relations and concepts which were formed by relating propositions in a fact corpus with science questions concepts. As compared to the query expansion baseline, their task driven relations and relation-based strategies proved to be more effective.

ISOFT, a hybrid QA system over linked data and text data had been developed by Park *et al*. (2015) [113]. They used Knowledge-Base based Question Answering (KBQA) and Information Retrieval based Question Answering (IRQA) approach to solve the complex questions. In order to avoid loss of meaning, they combined

important words. These combined words were compared with URIs, *i.e*., Uniform Resource Identifiers from a knowledge-base. Because of this method, the F-measure of their system was also improved.

An answer selection for community question answering called as QCRI (named after Quatar Computing Research Institute) had been developed by Nicosia *et al*. (2015) [111]. They applied a supervised machine learning approach considering the features like n-grams, text similarity, sentiment analysis, the presence of specific words, and the context of a comment. The proposed approach gave better results on Arabic in comparison to English.

Quan *et al*. (2015) combined multiple features for answer selection in community question answering and developed JAIST (named after Japan Advanced Institute of Science and Technology) [119]. Their system combined 16 features belonging to 5 groups to predict answer quality. Their system was able to identify good answers from the answer thread of community question answering. For English language, the model achieved the best results in terms of F-measure and accuracy.

Helena *et al*. (2015) proposed CICBUAPnlp which was a graph-based approach for answer selection in community question answering task [63]. For each answer class, their approach transformed the training set answers into graph-based representation. It contained morphological, lexical, and syntactic features. The system was able to construct several vectors of features and supported only English language. In order to calculate the accuracy of the system, two subtasks were taken. For subtask A an accuracy of 53.74 % was achieved while for subtask B it was 44.0%.

Stupina *et al.* (2016) proposed a QA system [144] which offered the approach to the determination of the question-answering system of giving an answer to the question based on a modified measure of one set to another inclusion. With relatively small knowledge-base, their system was able to give the correct answer in more than 50% of the cases.

Frederik *et al.* (2016) developed an HPI QA system (named after Hasso-Plattner Institute) in BioASQ 2016 [50]. The two main datasets on which the system relied were MEDLINE and UMLS (Unified Medical Language System). MEDLINE contained all the biomedical publications, whereas UMLS was a single knowledge-base that was created by combining various sources. UMLS was used in the proposed

system for named entity recognition. The mean average precision of their system was 0.434.

KSAnswer was developed by Hyeon-gu *et al.* (2016) [67]. This was a QA system of Kangwon National University and Sogang University in the 2016 BioASQ challenge. By using a cluster-based language model, their proposed system was able to retrieve candidate answer sentences. On the basis of the shallow semantic analysis, the system re-ranked the retrieved top-n sentences by using five different similarity models which were independent to each other.

A biomedical QA system based on public web services and ontologies was developed by Miguel *et al.* (2016) [97] known as WS4A. They used the existing web services for each step of finding answers. The information which was retrieved by the proposed system consisted of ontologies, concept identifiers, ancestors, and PubMed identifiers. One major limitation of the system was that its response time was very high and F-measure of the system was 0.24 only.

Wanyun *et al*. (2016) developed KBQA which is a learning QA over QA corpora and knowledge-bases [169]. The system supported factoid questions. The knowledge- base coverage was boosted 57 times by expanding predicates in RDF knowledge-base. They tested their system on QALD-5. The recall of the system was 0.50 and precision of 0.86.

Jun *et al*. (2016) proposed a Neural Generative Question Answering (GENQA) that can generate answers to simple factoid questions, based on the facts in a knowledgebase [75]. This model was based on encoder-decoder framework which was used for sequence to sequence learning. The GENQA model consists of interpreter, enquirer, answerer, and an external knowledge-base. The system was tested on the data from the web and an accuracy of 52% was achieved.

Xiong *et al*. (2016) worked on both visual and textual question answering [174]. They proposed a new framework DMN+ by making improvements in input and memory modules of existing Dynamic Memory Network (DMN). In order to provide the ability to answer visual questions, along with these changes they introduced a novel input module for images. The overall accuracy of the system was 28.79 with a mean error of 2.8 only.

Chandu *et al*. (2017) developed WebShodh, which was a code-mixed factoid QAS

for web [34]. They had demonstrated the system with two code-mixing languages which were Hinglish (a mixture of Hindi natural language and English natural language) and Tenglish (a mixture of Telugu natural language and English natural language). They had calculated the efficiency of their system by using MRR. For Hinglish, system's MRR was 0.37 whereas for Tenglish it was 0.32.

Mohamed (2017) have developed an automatic question answering system for the Arabic Quran [101]. They used Java, Microsoft Excel, Microsoft Access, and Lucene search engine for development of the proposed QA system. A version of the question answering system was built which used noun number patterns to process the number in Arabic questions and candidate answers, which enhanced the result set of answers by adding more words and meaning. A corpus of questions and its answers about the Holy Quran was used to test and compare baseline and enhanced versions of their Quran question answering system.

Kotaro *et al.* (2017) proposed FelisCatusZero, a world history essay question answering system for the University of Tokyo's entrance exam [80]. The essay answer generating algorithm of the system was based on extractive multi-document summarization. The eight phases, *i.e.*, question analysis, document retrieval, sentence extraction, sentence grouping, sentence ranking, answer candidate reduction, answer candidate ranking, and final answer selection formed the outline of the system.

Xiaoyi *et al.* (2017) had designed a semantic network theory to build up an intelligent answering system based on remote service framework [173]. The system provided the user with a synchronous and asynchronous communication, artificial intelligence and joint Q combined with considerable integration, convenience and practicality of remote answering environment. This system had been proposed mainly to facilitate the communication between the teachers and students of a university.

Nabeel and Saidah (2017) developed a question answering system supporting vector machine (SVM) method for hadith domain [108]. Apart from effective analysis of query need, SVM method reduced searching scope of Hadiths documents based on various subjects and question types using NLP methods. SVM provides more accurate answers than extracting answers using only similarity techniques such as CS and LCS.

A wikipedia based essay question answering system for university entrance examination had been proposed by Takaaki *et al.* (2017) [148]. The system generated

essays to answer the real examination questions for the admission to the Tokyo University. The proposed system supported English language.

Kwong and Chih (2017) proposed an automated QA system using a hybrid approach, a combination of the knowledge-based and text-based approaches [84]. This approach only required two SPARQLs (an RDF query language pronounced as “sparkle”) to retrieve the candidate answers from the ontology without defining any question pattern, and then used the topic model to find the most related candidate answers as the answers. They also investigated and evaluated different language models (unigram and bigram). Their results had shown that this QA system was able to perform beyond the random baseline and solve up to 44 out of 80 questions with MRR of 38.73%.

In order to promote the tourism service more automated and intelligent, Hua and Shi-Zheng (2017) proposed a design of intelligent tourism QA system based on semantic web [66]. By greatly improving the semantics of sentence, the algorithm posed in the system provided a new method for question matching as a fusion similarity algorithm based on literal similarity and semantic similarity.

An automatic web-based QAS for e-learning had been developed by Waheeb and Babu (2017) [167]. They had developed technique to detect the type of a question, based on which the proper technique for extracting the answer was used. The system returned only blocks or phrases of data containing the answer rather than full documents.

In order to converse over linked question answer pairs with a knowledge graph, a complex sequential QAS had been proposed by Amrita *et al.* (2018) [14]. Through a labor intensive semi-automatic process, involving in-house and crowd sourced workers, they created a dataset containing around 200K dialogs with a total of 1.6M turns. Further, unlike existing large scale QA datasets which contain simple questions that can be answered from a single tuple, the questions in their dialogs required a larger subgraph of the knowledge graph.

Gupta *et al.* (2018) had proposed a framework for linguistically driven question generation and neural based question answering [57]. They had proposed a linguistically motivated technique for codemixed question generation (CMQG) and a neural network based architecture for code-mixed question answering (CMQA). For evaluation, they manually created the code-mixed questions for Hindi-English

language pair. They evaluated their system on CMMQA dataset and obtained the EM and F1 score of 40.50% and 53.73% respectively.

Mohnish *et al.* (2018) proposed a framework called EARL (Entity And Relation Linker), which performs entity linking and relation linking as a joint single task [46]. EARL used a graph connection based solution. EARL implemented two different solution strategies. The first strategy was a formalization of the joint entity and relation linking tasks as an instance of the Generalised Travelling Salesman Problem (GTSP). The second strategy used machine learning in order to exploit the connection density between nodes in the knowledge graph.

Architecture for Chinese question answering system had been proposed by FENG *et al*. (2018) [48]. This proposed Chinese Question Answering System in Medical Domain (CQASMD) provided useful medical information for users. A large medical knowledge-base with more than 300 thousand medical terms and their descriptions was firstly constructed to store the structured medical knowledge data, and classified with the FastText model. Furthermore, a Word2Vec model was adopted to capture the semantic meanings of words, and the questions and answers were processed with sentence embedding to capture semantic context information.

A syntactic and semantic multi-agent based QAS for collaborative e-learning had been proposed by Abderrazzak *et al.* (2018) [1]. This QAS helped learners to find the best answers to their questions and also helped tutors to answer questions asked by their students in an e-learning environment.

A visual question answering system which uses an explicit trained attention model had been proposed by Vasileios *et al*. (2018) [162]. They used attention-oriented word embeddings that increased the efficiency of learning common representation spaces. The Visual7W dataset which was used, was the only dataset that provided visual attention ground truth information.

A QA based framework for information extraction, *i.e.*, QA4IE had been proposed by Lin *et al*. (2018) [89]. Cross-sentence tuples, limited relation types and informal relation specifications (*e.g.*, free-text based relation tuples) were the challenges which had been overcome in QA4IE. Based on the framework, they developed a large IE benchmark with high quality human evaluation.

Lekshmi *et al.* (2018) proposed a question answering system which used a combined approach using word sense disambiguation and semantic role labelling [87]. The proposed model of work was factoid sense based question generation system. They had used Lesk algorithm for word sense disambiguation and Senna tool for semantic role labelling. Using deep syntax and semantics analysis, they had extracted an answer from the given question. Hobbs algorithm resolved co-reference problem generated in answer extraction. The answer extraction accuracy of the system was 85%.

Muhammad and Noman (2018) proposed convolutional neural network, a text classification model for open domain question answering system. Neural network model was trained on top of word embedding [106]. Softmax layer was applied to calculate loss and mapping of semantically related words. The had used F1 score to evaluate their proposed system. Out of three experiments, the highest F1 score achieved by their proposed system was 69%.

A Multi-domain Multi-lingual Question Answering (MMQA) framework for English and Hindi language had been proposed by Gupta *et al.*, (2018) [56]. They had developed a deep learning based model for classifying an input question into the coarse and finer categories depending upon the expected answer. The answers were extracted through similarity computation and subsequent ranking. For factoid question, they obtained an MRR value of 49.10% and for short descriptive questions they obtained a BLEU score of 41.37%.

The question and answering systems discussed above were proposed by using various approaches. The subsection below identifies the major research and development activities in UNL for information retrieval, search engines, and question answer systems.

2.1.1 UNL Based Research Activities for Information Retrieval, Search Engines, and QA Systems

For the agriculture domain in Marathi, Hindi, and English language, an interlingua meaning-based search engine had been proposed by Surve *et al*. (2004) [147]. This system was called as ‘Agro-Explorer’. The proposed system used focused crawler, HTML parser, enconverter, preprocessor, indexer, search module, deconvertor, and post-processor. The focused crawler was responsible to crawl the web and fetch documents. The design of the page was maintained by HTML parser. The enconverter

converted the text extracted from the HTML corpus into its equivalent UNL representation. The preprocessor was responsible to convert the UNL corpus into an intermediate format. The responsibility of indexer was to take the UNL corpus and generate an inverted index on it. The user query was converted to UNL. The preprocessor that was used for indexing, also, preprocessed the user query after conversion to UNL. After this, it was then passed on to the search module. In order to perform a graph-based search on the UNL query, UNL index was used by the search module. Those UNL documents that match with the query were returned by the search module. Now, these UNL documents were converted into the language of the user's choice by the deconverter. The translated text and the design of the page stored by HTML parser was merged by the post-processor to form an HTML page which was presented to the user.

Shukla *et al*. (2004) proposed a question answering system and suggested two major tasks for implementing it [134]. The first task was called as document processing which involved conversion to the UNL expression from the equivalent document. For the document, its complete set of meaning based predicates was obtained by processing the document through a UNL parser. The second task was the answer extraction after processing the query. The proposed system used Saarthak, *i.e*., a multilingual HPSG parser which outputs a pseudo-UNL expression corresponding to the proposed answer template.

For question answering system for language-independent semantics, UNL as a tool was proposed by Mukerjee *et al*. (2005) [107]. In order to search for the answer to a question, a structured matching approach was used. An answer template was built that represented the form of the answer corresponding to the question. HPSG parser took this template as an input. Now, corresponding to the question and the answer template, HPSG parser outputted a pseudo-UNL expression. The pseudo-UNL expression was subsequently subjected to a structure matching with the UNL document (the corpus knowledge-base).

Karande (2007) proposed a multilingual search engine using UNL [78]. Before building index terms or list of keywords for implementing multilingual search engine using UNL, the conversion of contents from any language to UNL was required. The spider was responsible for input to the convertor, which were pages in the native language. The convertor converted these native language pages into UNL. The

language of web pages was identified by the convertor and web page was sent to that corresponding language server. The native language page was translated into UNL by language server. The convertor also converted the query into UNL just before the time of searching for information. Now, since query as well as index terms were available in UNL form, all searching operations were performed on UNL. Finally, the result was converted into the native language in which the query was asked.

Cardenosa *et al*. (2009) had proposed an interlingual information extraction as a solution for multilingual QA systems [33]. The system launched a query in any language against a language-independent representation of the document set using a general-purpose UNL interlingua and received a precise response. The architecture of the proposed system had three components, *i.e.*, query pre-processing module, UNL query converter, and search engine. The system was tested on UNL-EOLSS corpus provided by UNDL foundation. The accuracy of the system was 82.6% because the system answered 62 of the 75 questions correctly.

A 'LOOK4' system have been proposed by Avetisyan and Avetisyan (2010) to enhance online search results with the help of UWs [17]. They had leveraged on a few existing resources such as WordNet semantic ontology, online services given by major search engines, and flexibility of UNL used in representing semantic relations among concepts. The overall average accuracy of 'LOOK4' system was 80.6%.

Goel (2016) proposed an information retrieval system using UNL [54]. The system worked on factoid questions with the help of predefined mappings of universal relations with query words. The system was tested on EOLSS corpus provided by UNDL foundation. A total of 45 manually created questions were framed on the article "Water supply and agriculture" in EOLSS corpus. The system was able to answer 37 questions out of 45. The overall accuracy of the system came out to be 82.22%.

After doing an exhaustive survey of different question answering systems, it was observed that these systems differs in various key metrices like corpus, evaluation metrics, domain and depth of explanation of their working components. Out of all the QA systems surveyed, there were very few QA systems in which working algorithms of all their components were explained. Table 2.1 below shows the comparison of important metrics of various QA systems.

Table 2.1: Comparison of QA System's Key Metrics

	QA System Key Metrics				
QA Details / **QA Research**	**Corpus Used for Testing**	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**
MMQA (Gupta *et al.*, 2018)	500 articles from web in Tourism, History, Diseases, Geography, Economics, and Environment domain.	MRR and BLEU Score	MRR = 49.10% BLEU Score = 41.37%	Open	×
Complex Sequential Question Answering (Amrita *et al.*, 2018)	Wikidata dump	Precision and Recall	Precision = 6.3% Recall = 18.4%	Open	×
EARL (Mohnish *et al.*, 2018)	LC-QuAD and QALD-7	Accuracy	For LC-QuAD = 0.65 For QALD-7 = 0.57	Open	×
A multi-agent based Question Answering System (Abderrazzak *et al.*, 2018)	Not specified	Not specified	Not specified	Not specified	×

Visual Question Answering using Explicit Visual Attention(Vasileios *et al.*, 2018)	Visual7W	Accuracy	0.659	Open	×
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**
CQASMD (FENG *et al.*, 2018)	Not specified	Not specified	Not specified	Closed	×
QA4IE (Lin *et al.*, 2018)	SQuAD	F1 Score	72.5	Open	×
An automatic question answering system for the Arabic Quran (Mohamed, 2017)	Holy Quran (Surat Al-Fatiha and Al-Baqarah Chapters)	Percentage Right Answers	92.9%	Closed	×
WebShodh (Chandu *et al.*, 2017)	Random Questions (100 Hinglish + 100 Tenglish)	MRR	Hinglish (0.37) Tenglish (0.32)	Open	×
FelisCatusZero (Kotaro *et al.*, 2017)	Few mock exams of the University of Tokyo's entrance exam	Manual	10 out of 20	Open	×
Intelligent Question Answering System Based on Remote Service Framework (Xiaoyi *et al.*, 2017)	Real time chat room exchanges, online discussion forums	Not specified	Not specified	Open	×
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

Question Answering System Supporting Vector Machine Method for Hadith Domain (Nabeel and Saidah, 2017)	Hadith corpus (Pray and Fasting subjects)	F-Measure	80%	Closed	✕
Wikipedia Based Essay Question Answering System for University Entrance Examination (Takaaki *et al.*, 2017)	QA task of NTCIR (NII Testbeds and Community for Information access Research) QA Lab 3	ROUGE-1 Mean	0.584	Open	✕
A Hybrid Question Answering System based on Ontology and Topic Modeling (Kwong and Chih, 2017)	80 questions with answers (20 questions per categories) from the textbooks as the gold standard	MRR	38.73%	Open	✕
Design of Intelligent Tourism Question Answering System Based on Semantic Web (Hua and Shi-zheng, 2017)	2000 standard question on Chongqing tourism attractions, tourism traffic and tourism catering	Precision	86%	Closed	✕
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

An automatic web-based question answering system for e-learning (Waheeb and Babu, 2017)	World Wide Web	F-Measure	80%	Open	✗
Information Retrieval System using UNL for Multilingual Question Answering (Goel and Kumar, 2016)	UNL-EOLSS	Accuracy	81.5%	Open	✓
Question-answering system (Stupina *et al.*, 2016)	Following 3 books: Life Science concepts for middle school, Biology, Earth Science Concepts For Middle School.	Accuracy	53.6%	Open	✓
HPI Question Answering System (Frederik *et al.*, 2016)	Document provided by BioASQ 2016	Mean Average Precision	0.434	Closed	✗
KSAnswer (Hyeon-gu *et al.*, 2016)	Document provided by BioASQ 2016	Mean Average Precision	0.3752	Closed	✗
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

WS4A (Miguel *et al*., 2016)	Document provided by BioASQ 2016	F-Measure	0.24	Closed	✗
KBQA (Wanyun *et al*., 2016)	QALD-5	Precision and Recall	Precision: 0.86 Recall: 0.50	Open	✓
GENQA (Jun *et al*., 2016)	World Wide Web	Accuracy	52%	Open	✗
A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering (Wang and Nyberg, 2015)	Text REtrieval Conference (TREC) QA track (8-13) data	MRR	0.7913	Open	✗
Question Answering system using vector space model (Hartawan and Suhartono, 2015)	Data comes from 2 ministers from Indonesia; they are Minister of Education & Culture and Minister of Tourism & Creative Economy Culture.	F-Measure	0.580	Open	✗
ISOFT (Park *et al*., 2015)	QALD-5 test dataset	F-Measure	0.33	Open	✗
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

Learning Knowledge Graphs for Question Answering through Conversational Dialog (Ben *et al.*, 2015)	Identity, WordNet, PPDB, KNOWBOT, LEAVE-ONE-OUT	Percentage Correct	43.7%	Open	✗
YodaQA (Baudis, 2015)	Public QA benchmark from TREC 2001 and 2002, QA tracks with regular expression answer patterns and extended by questions asked to a YodaQA predecessor by internet users via an IRC interface.	Recall	79.3%	Open	✗
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

QCRI (Nicosia *et al*., 2015)	SemEval-2015 Task 3 "Answer Selection in Community Question Answering"	F-Measure	English Subtask A: 53.74 (English) English Subtask B: 53.60 (English) English Subtask A: 53.74 (Arabic) English Subtask B: 53.60 (Arabic)	Open	×
JAIST (Quan *et al*., 2015)	SemEval-2015 Task 3 "Answer Selection in Community Question Answering"	F-Measure	English Subtask A : 57.18	Open	×
CICBUAPnlp (Helena *et al*., 2015)	SemEval-2015 Task 3 "Answer Selection in Community Question Answering"	F-Measure	English Subtask B : 38.8	Open	×
CASIA@V2 (Shizhu *et al*., 2014)	QALD-4 dataset	F-Measure	0.36	Open	×
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

Al-Bayan (Heba *et al*., 2014)	Holy Quran	Accuracy	85%	Closed	✗
Forst (Kotaro *et al*., 2014)	Japanese textbooks and Wikipedia	ROUGE Scores	ROUGE-1 Score: 0.125 ROUGE-2 Score: 0.062 ROUGE-L Score: 0.097	Open	✗
Knowledge-Based Question Answering as Machine Translation (Junwei *et al*., 2014)	WEBQUESTIONS CORPUS (3,778 questions)	Accuracy and Precision	32.5% (Accuracy) 73.2% (Precision)	Open	✓
CMU Multiple-choice Question Answering System (Di *et al*., 2014)	NTCIR-11 QA Lab evaluations	Accuracy	51.6%	Open	✗
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

A Natural Language Question Answering System in Malayalam Using Domain Dependent Document Collection as Repository (Pragisha and Reghuraj, 2014)	The system was tested with 100 different questions included in the four classes of question words. Ten documents in the domain of Indian history and geographical features of India were used for the testing.	Accuracy	81%	Closed	✓
Watsonsim (Sean *et al.*, 2014)	Jeopardy	Accuracy	18%	Open	✗
QASYO (Abdullah *et al.*, 2011)	YAGO ontology version 2008-W40-2 (100 questions)	Accuracy	91%	Open	✗
LOOK4 System (Avetisyan *et al.*, 2010)	World Wide Web	Accuracy	80.6%	Open	✗
Interlingual information extraction as a solution for multilingual QA systems (Cardenosa *et al.*, 2009)	UNL-EOLSS	Accuracy	82.6%	Open	✗
	Corpus Used for Testing	**Evaluation Metrics**	**Evaluation Metrics Value**	**Domain**	**Working Algorithm of all components explained**

AGRO-EXPLORER System (Surve, 2004)	Agro Explorer Corpus	Accuracy	73.52%	Closed	✗
NSIR (Radev *et al*., 2002)	TREC-8 (200 questions)	MRR	0.974	Open	✗
LASSO (Moldovan *et al*.,1999)	TREC-8 (200 questions)	Accuracy	55.5% (short answers) 64.5% (long answers)	Open	✗
Lunar (Woods, 1973)	Chemical analysis data on lunar soil and rock composition	Accuracy	78.2%	Restricted	✗
Baseball (Green *et al*., 1963)	Baseball games played in the American league over one season.	Accuracy	84.38%	Restricted	✗
Ask Jeeves (Online)	World Wide Web	Not specified	Not specified	Open	✗
ASKMSR (Michele *et al*., 2002)	TREC-9	MRR	0.507	Open	✗

The QA systems studied in Table 2.1 were also analyzed for the some important characterstics like online availability, multilinguism support, ability to extend the support to other languages, ability to integrate other NLP applications, and availability of source code in which these QA systems differ. Table 2.2 depicts the comparison of these features missing/present in various QA systems.

Table 2.2: QA System's Features Comparison

	QA System Features

QA Details / QA Research	Available Online	Supports Multi Language	Can be extended to support other languages	Can integrate other NLP applications	Is Source Code Available
MMQA (Gupta *et al.*, 2018)	✗	✓ (English and Hindi)	✗	✗	✗
Complex Sequential Question Answering (Amrita *et al.*, 2018)	✗	✗ (English)	✗	✗	✗
EARL (Mohnish *et al.*, 2018)	✗	✗ (English)	✗	✗	✗
A multi-agent based Question Answering System (Abderrazzak *et al.*, 2018)	✗	✗ (English)	✗	✗	✗
Visual Question Answering using Explicit Visual Attention (Vasileios *et al.*, 2018)	✗	✗ (English)	✗	✗	✗
CQASMD (Feng *et al.*, 2018)	✗	✗ (Chinese)	✗	✗	✗
QA4IE (Lin *et al.*, 2018)	✗	✗ (English)	✗	✗	✗
WebShodh (Chandu *et al.*, 2017)	✓	✓ (Hindi, English, Telugu)	✗	✗	✗
An automatic question answering system for the Arabic Quran (Mohamed, 2017)	✗	✗ (Arabic)	✗	✗	✗
	Available Online	Supports Multi Language	Can be extended to support other languages	Can integrate other NLP applications	Is Source Code Available

FelisCatusZero (Kotaro *et al.*, 2017)	×	× (Japanese)	×	×	×
Intelligent Question Answering System Based on Remote Service Framework (Xiaoyi *et al*., 2017)	×	× (English)	×	×	×
Question Answering System Supporting Vector Machine Method for Hadith Domain (Nabeel and Saidah, 2017)	×	× (English)	×	×	×
Wikipedia Based Essay Question Answering System for University Entrance Examination (Takaaki *et al*., 2017)	×	× (English)	×	×	×
A Hybrid Question Answering System based on Ontology and Topic Modeling (Kwong and Chih, 2017)	×	× (English)	×	×	×
Design of Intelligent Tourism Question Answering System Based on Semantic Web (Hua and Shi-zheng, 2017)	×	× (Chinese)	×	×	×
An automatic web-based question answering system for e-learning (Waheeb and Babu, 2017)	×	× (English)	×	×	×
Question-answering system (Stupina *et al*., 2016)	×	× (English)	×	×	×
Information Retrieval System using UNL for Multilingual Question Answering (Goel and Kumar, 2016)	×	× (Punjabi)	×	×	×
HPI Question Answering System (Frederik *et al*., 2016)	×	× (English)	×	×	×
	Available Online	**Supports Multi Language**	**Can be extended to support other languages**	**Can integrate other NLP applications**	**Is Source Code Available**

KSAnswer (Hyeon-gu *et al*., 2016)	×	× (English)	×	×	×
WS4A (Miguel *et al*., 2016)	×	× (English)	×	×	×
KBQA (Wanyun *et al*., 2016)	×	× (English)	×	×	×
GENQA (Jun *et al*., 2016)	×	× (Chinese)	×	×	×
A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering (Wang and Nyberg, 2015)	×	× (English)	×	×	×
Question Answering system using vector space model (Hartawan and Suhartono, 2015)	×	× (English)	×	×	×
ISOFT (Park *et al*., 2015)	×	× (English)	×	×	×
Learning Knowledge Graphs for Question Answering through Conversational Dialog (Ben *et al*., 2015)	×	× (English)	×	×	×
YodaQA (Baudis, 2015)	×	× (English)	×	×	✓
QCRI (Nicosia *et al*., 2015)	×	✓ (Arabic, English)	×	×	×
JAIST (Quan *et al*., 2015)	×	× (English)	×	×	×
	Available Online	**Supports Multi Language**	**Can be extended to support other languages**	**Can integrate other NLP applications**	**Is Source Code Available**

CICBUAPnlp (Helena *et al*., 2015)	×	× (English)	×	×	×
CASIA@V2 (Shizhu *et al*., 2014)	×	× (English)	×	×	×
Al-Bayan (Heba *et al*., 2014)	×	× (Arabic)	×	×	×
Forst (Kotaro *et al*., 2014)	×	× (Japanese)	×	×	×
Knowledge-Based Question Answering as Machine Translation (Junwei *et al*., 2014)	×	× (English)	×	×	×
CMU Multiple-choice Question Answering System (Di *et al*., 2014)	×	× (English)	×	×	×
A Natural Language Question Answering System in Malayalam Using Domain Dependent Document Collection as Repository (Pragisha and Reghuraj, 2014)	×	× (Malayalam)	×	×	×
Watsonsim (Sean *et al*., 2014)	×	× (English)	×	×	×
QASYO (Abdullah *et al.*, 2011)	×	× (English)	×	×	×
LOOK4 System (Avetisyan *et al*., 2010)	×	× (English)	×	×	×
Interlingual information extraction as a solution for multilingual QA systems (Cardenosa *et al*., 2009)	×	× (English)	×	×	×
AGRO-EXPLORER System (Surve, 2004)	×	✓ (English, Hindi, Spanish, Marathi)	×	×	×
	Available Online	**Supports Multi Language**	**Can be extended to support other languages**	**Can integrate other NLP applications**	**Is Source Code Available**

NSIR (Radev *et al*., 2002)	✗	✗ (English)	✗	✗	✗
LASSO (Moldovan *et al*.,1999)	✗	✗ (English)	✗	✗	✗
Lunar (Woods 1973)	✗	✗ (English)	✗	✗	✗
Baseball (Green *et al*., 1963)	✗	✗ (English)	✗	✗	✗
Ask Jeeves (Online)	✓	✓	✗	✗	✗
ASKMSR (Michele *et al*., 2002)	✓	✗ (English)	✗	✗	✗

After analysing the key metrics and features (shown in Table 2.1 and Table 2.2), it has been observed that there is an immense need of QA system which should be available online, supports multilinguism, scalable for multiple languages and having the ability to integrate with other UNL based NLP applications. Since, UNL is a major framework to implement multilingual systems, in the next section the research work on development of UNL resources has been documented and analysed.

2.2 Research Activities in UNL

During the last couple of years, in the field of NLP, UNL has been an area of interest for researchers all over the world for the development of multilingual systems.

UNL aims at coding, storing, propagating and retrieving information independently of the original language in which it was expressed. UNL has been exploited in a number of other domains such as text summarization, sentiment analysis, multilingual document generation, semantic reasoning including transition in the field of NLP.

Sérasset and Boitet (2000) called UNL as the future *"*html of the linguistic content*"* [130]. The authors built the French deconverter on the basis of modular architecture. The deconversion process was split into two steps such as transfer and generation. In transfer step, the UNL graph was validated, localized and then lexically transferred. In generation phase, Unique Multilevel Concrete (UMC) structure was obtained from Unique Multilevel Abstract (UMA) structure.

Sornlertlamvanich *et al.* (2001) performed the Thai lexical semantic annotation by UWs [141]. The authors proposed the process for word sense disambiguation, UW annotation and word extraction. A computational method to identify an appropriate UW to annotate a Thai word was also proposed by the authors.

Bhattacharyya (2001) proposed multilingual information processing through UNL [22]. The system was able to perform sentence level encoding of English, Hindi and Marathi into the UNL form and then decoding of this information to Marathi and Hindi. As such, a way had been created by the system of translation which was semi-automated from Hindi to Marathi and also from English to Hindi and Marathi.

Unparalleled features of UNL taking conclusions from Brazilian Portuguese-UNL enconverting task was analyzed by Martins (2002) [95]. They suggested that UNL should not be treated as an interlingua, rather it should be considered as a source and a target language because of the flexibility of UNL given by enconversion process.

An enconversion tool from Tamil was proposed by Dhanabalan *et al*. (2002) [42]. An existing morphological analyzer of Tamil was used by their system to extract the morphological features of an input sentence. A specially designed parser was employed by them so as to perform syntactic functional grouping. The enconversion rules which were written for Tamil language drive the entire enconversion process.

The lexical items, called UWs, have been founded by Lafourcade and Boitet (2002) which were not connected to lemmas [86]. The authors used the concept of conceptual vectors for the disambiguation process. The vectors were attached to the tree and then propagated up to the root. The vector mutual activation was induced by performing backtracking towards the leaves.

Using UNL representation, text clustering has been performed by Choudhary and Bhattacharyya (2002) [35]. Using UNL, featured vectors have been generated by them. The proposed system used SOMs (Self Organizing Maps) for clustering. They

concluded that for feature vector generation, frequency based methods did not perform well as compared to UNL method.

Within a UNL framework in order to represent speech control mechanisms, a VoiceUNL was proposed by Tomokiyo and Chollet (2003) [151]. Speech to Speech Machine Translation (SSMT) was also supported by this system. Although the UNL design was able to support written texts but still the feasibility of augmenting the UNL attributes tags to complement SSMT processing had been explored by the developers.

A deconverter for Tamil language was proposed by Dhanabalan and Geetha (2003) [41]. A framework for morphological generation, selection of word, syntactic generation and natural collocation which was mandatory for the formation a sentence was provided synchronously by the proposed system which acted as a language-independent generator. In order to get natural language sentences from the UNL structure, linguistic based and language specific deconversion rules were used by the proposed system.

Martins (2003) addressed the color categorization problem from the multicultural knowledge representation perspective [93]. The author considered three different network like structured, knowledge representation formalisms such as Conceptual Graphs (CG), Resource Description Framework (RDF) and UNL as candidate alternatives. Martins concluded that UNL is better than the others and it can cope with multilingualism and cross-cultural color representation schemes.

Ribeiro *et al.* (2004) developed a Portuguese-UNL dictionary [123]. The authors used a semi-automatic process to build the Portuguese-UNL dictionary by reusing the existing resources such as UNLKB and WordNet.PT (a lexical database developed under the EuroWordNet framework). For this, the similarities between UNL and lexical semantic relations of WordNet.PT were explored.

For Chinese language, a UNL based deconverter was proposed by Shi and Chen (2005) [133]. The unavailability of the source code and its slow speed, and challenges in writing the rules were few challenges of ‘DeCo’ tool (provided by UNDL center) which had been highlighted by them. So, a new deconverter for Chinese language was proposed by these developers to address these issues.

The use of knowledge acquisition process in intelligent management systems was explored by Bueno *et al*. (2005) [29]. The knowledge engineering suite which was

used to support the construction of ontologies was the result of their efforts. They used both a knowledge representation technique called DCKR (Dynamically Contextualized Knowledge Representation) to organize knowledge, and psychoanalytic studies, focused mainly on Lacan and his language theory to develop a methodology called mind engineering.

Inputs from a human expert were required by universal parser and 'EnCo' tool developed by UNDL foundation. Martins *et al*. (2005) analyzed that the performance of the human expert is not up to the mark and the human expert is not frequently available [94]. Therefore, this issue was addressed by the authors by proposing a system called 'HERMETO'. The system converted Brazilian Portuguese and English to UNL. The proposed systems semantic and high level syntactic grammar ability made it more user friendly.

For UNL based machine translation system, complex case structure of Bengali language's computational analysis was presented by Dey and Bhattacharyya (2005) [40]. For Bengali language, analysis and generation rules drives the 'EnCo' and 'DeCo' tools. The case structure of 'kaaraks' which were used in Bengali language formed the base of these rules.

For English language machine translation system which was UNL based, Prepositional Phrase (PP) attachment problem was explored by Mohanty *et al.* (2005) [102]. Linguistic analysis for English language's six common prepositions, namely, *"in", "for", "to", "from", "with",* and *"on",* was performed by them. For the 'EnCo' tool, a rule base and lexicon was enriched because of this analysis results.

In order to develop a UNL based machine translation system, a SRS (Semantically Relatable Sequence) based method was used by Mohanty *et al.* (2005) [103]. The source language was analyzed using semantic graphs. The target language text was generated using these graphs. The performance of the system was tested on the Penn Treebank and the proposed system indicated effective and promising results.

Blanc (2005) proposed French deconverter and French enconvertor [24]. He also did the integration of 'Ariane-G5' to this. In machine translation systems, 'Ariane-G5' was used as a generator. In the suggested system, there were two steps in which enconversion happened. As a part of first step, the production of the representation of French text's meaning was analyzed in the form of a dependency tree. From this

dependency tree, the transfer of structural and lexical to an equivalent UNL graph formed the second step of the proposed enconversion process.

Boguslavsky *et al.* (2005) proposed a multi-functional linguistic processor 'ETAP-3' as an extension of 'ETAP' machine translation system to a UNL based machine translation system [25]. The enconversion of NL texts into UNL is carried out by means of a multi-functional linguistic processor 'ETAP-3'. The resolution of ambiguity in linguistic units was ensured by the linguistic knowledge-base of 'ETAP-3'.

Choudhury *et al.* (2005) proposed a framework to convert Bangla to UNL. They also proposed a procedure for the construction of Bangla to UNL dictionary [36]. They had elaborated the concepts and building block of UNL in their work. The entire UNLization details like analysis dictionary specifications, transformation rules *etc.* had been described by them.

Lafourcade (2005) uses ant colony algorithm to develop the system for semantic analysis and fuzzy UNL graphs for enconversion process [85]. They presented an extension of the UNL graph structure aiming at handling lexical and relational ambiguities. On that intermediate structure, they applied ant algorithm propagation of conceptual vectors and other constraints.

For communication among different cultures and languages, UNL has been explored as a facilitator by Jiang *et al*. (2005) [74]. Geopolitical interdependence and markets current globalization trends had given rise to several critical problems. These problems were addressed and solved by the system designed by them. In a distributed environment, people across the world with different cultural and linguistic backgrounds constructed the UNL knowledge-base by the facilitation provided by their system.

For knowledge annotation a UNL XML (Extended Markup Language) model had been proposed by Cardeñosa *et al*. (2005) [32]. For data mining of complex information, development of new annotations or labels had been suggested by them. The current analysis techniques which were text-based had limitations. In order to overcome these limitations, usage of UNL had been proposed by them.

Hajlaoui and Boitet (2005) had proposed a pivot XML based architecture for multilingual, multiversion documents through UNL [58]. In a first stage, they splitted

the UNL-XML document in several monolingual documents. Each document contained the text in a particular language, plus the corresponding UNL graphs, and were modified independently.

Derived from UNL, UCL (Universal Communication Language) had been proposed by Montesco and Moreira (2005) [105]. UCL had been used for communication among humans and among different software agents. In order to ease the integration of UCL with the internet, XML had been used to define UCL.

In order to improve case retrievals in case based reasoning systems, use of semantic information had been proposed by Iyer and Bhattacharyya (2005) [71]. They also proposed a UNL based system to improve the precision of retrieval. The semantic knowledge hidden in the words of a problem sentence was taken into account in the proposed method. To find the semantic similarity between two concepts, WordNet was used by the system.

Manati deconversion model had been introduced by Pelizzoni and Nunes (2005) [114]. For human-aided machine translation system of Portuguese-Brazilian sign language, this model acted as a UNL mediator. Unlike the higher-order and object oriented programming which provided definition for user friendly primitives, constraint programming reduced the search. This constraint programming formed the basis of this system.

To use UNL online, a web platform called as 'CELTA' was proposed by Bértoli *et al*. (2005) [21]. By using UNL, a multilingual business-to-business platform was showcased by them. The implementation of this platform adopted an incremental method of using UNL resources to avoid interference from a technology that was not yet mature. In order to isolate the platform from problems related to UNL development, an interface was provided by UNL web service.

For linking Spanish-UNL dictionary's UWs, WordNet was used by Iraola (2005) [70]. Their system had enriched UWs with semantic information. Focusing on a subset of the Spanish-UNL dictionary, namely on the substantives it contained, the work had consisted in automatically enriching the UW associated with each substantive with the semantic information required to link the UW to the universal word system.

By using Marathi, Hindi, and English WordNets, multilingual lexicon's automatic generation for Marathi, Hindi, and English have been done by Verma and

Bhattacharyya (2005) [166]. UNL's knowledge-base, word sense disambiguator, and an inferencer along with WordNet were used by the system.

Feature extractor based on statistical techniques had been used for UNL relations classifier's training by Nguyen and Ishizuka (2006) [110]. In addition to the common used features, they also proposed a new feature that reflected the actual semantic relation of two phrases independent on words in between. When evaluated on a dataset (provided by UNDL foundation), their system gave an accuracy of 79%.

To transfer information from and to rural communities, a web based multimedia and multilingual portal for agriculture was developed by Ramamritham *et al*. (2006) [121]. This was an improved and enhanced version of 'Agro-Explorer' which was called as 'aAQUA'. Information retrieval techniques like meaning based search, intelligent caching, and intermittent synchronization with offline access, intelligent caching *etc.* and some novel database systems were being used by this 'aAQUA'.

Within the UNL framework, the concept of universal digital library which was language-independent was proposed by Ansary *et al*. (2006) [10]. As a proof of concept they implemented UNL based LIS (Library Information System). Access to the books of different languages was provided by UNL LIS into the native language of the user.

For a project called Survival Translator (SurviTra), English-French-Hindi-UNL resources were built by Boitet *et al.* (2007) [26]. This project was a web service. If English or any other common language was not an option, then this web service was used to help an Indian helper to communicate with a French visitor. A dictionary along with a phrasebook was equipped with this chat web service which was bilingual in nature.

For English to UNL conversion, Jain and Damani (2009) had employed a lexicalized probabilistic parser [73]. For a given English sentence to create phase structure tree and typed dependency tree, this parser was used. They measured the accuracy of the system by computing the BLEU score on the Hindi sentences generated from the UNL. On 60 sentences taken from a real life agricultural corpus, they achieved a BLEU score of 0.26 compared to a BLEU score of 0.33 for the manually generated UNLs.

By using WordNet, a strategy to build UW dictionary was proposed by Bekios *et al*. (2007) [19]. For creating meaning based restrictions of UWs, they proposed six rules. For the UWs that were being created, a suitable semantic restriction was produced by the system. The rule base requires 45 inputs which were given in the form of each WordNet word. It had the set of lexical relations and senses of that word.

The successful development and testing of a UNL based Arabic machine translation system had been done by Adly and Alensary (2009) and Bueno *et al*. (2005) [3][29]. For the machine translation system for the tools, Arabic generation grammar was created in this system.

Interlingua elucidation of texts has been proposed by Rouquet and Nguyen (2009) [124]. In large collections of images which were accompanied by texts, ways to allow multimodal multilingual search had been explored by them. For elucidation, the UNL UWs and domain ontology was used. Their lexical resource consisted of bilingual dictionaries from several languages to UNL, 200000 UWs, and WordNet.

By using WordNet ontology, UW dictionaries unification was proposed by Boudhh and Bhattacharyya (2009) [27]. Extending of UW dictionary to U++ UW dictionary with the help of WordNet ontology was proposed by them. To identify semantically similar contexts, they used the concept of similarity measures [27].

The enconversion system for Bangla language had been proposed by Mridha *et al.* (2010) [99]. To obtain the primary suffixes and the roots of Bangla words, morphological analysis was done. They had tested their proposed system with few sample sentences of Bangla language available at Russian UNL language server.

With the help of UWs, web search results can be enhanced. A system called 'LOOK4' was proposed by Avetisyan and Avetisyan (2010) for enhancing these results [17]. Some existing resources such as UNL's flexibility for representing semantic relations among concepts, some major search engines web based services, and WordNet's meaning based ontology had been used by them.

Kumar and Sharma (2012, 2013) proposed an enconversion and deconversion system for the conversion of Punjabi language to UNL and vice versa [83][82]. However, their system uses adhoc rules and is limited to only particular set of corpus. As per the recommendation provided by Uchida (1987), Dave and Bhattacharyya (2001), Dey

and Bhattacharyya (2005), the semantic, syntactic, and morphological information of Punjabi language forms the basis of creation of enconversion rules database [154][39][40].

For Punjabi language, development of IAN/UNLization module had been done by Agarwal (2013) [9]. The development of Punjabi's EUGENE/NLization module had been done by Verma and Kumar (2013), Singh (2013) [163][135].

Navaneethakrishnan *et al.* (2014) built a language independent discourse parser using UNL [109]. The discourse parser constructed Rhetorical Structure (RS) trees to identify complex discourse structures. The discourse parser constructed RS trees using high-level semantic features inherited from the UNL.

Mridha *et al.* (2014) used UNL and proposed a new approach to address the problem of semantic ambiguity of Bangla root words [100]. They had used insights from linguistics towards solving this problem. Also, the usefulness of automatic extraction of features for words in the dictionary became evident through their work.

Jadhav and Bhattacharyya (2014) had proposed an unsupervised rule-based approach using deep semantic processing to identify only relevant subjective terms [72]. In the proposed approach, UNL graph had been generated for the input text. The rules were applied on the graph to extract relevant terms. The sentiment expressed in these terms was used to figure out the overall sentiment of the text.

Ali *et al.* (2015) developed the word dictionary of Bangla vowel ended roots for first person for UNL [12]. The proposed entries were to be used to combine with their inflexions to produce verbs, and hence these verbs could be used for conversion of native language sentences into the UNL expressions. They provided the format of vowel ended roots along with their alternatives based on the framework of UNL provided by the UNL center of the UNDL foundation.

Multilingual acquiring of e-content definition based on UNL was proposed by Sathiyamurthy *et al.* (2015) [126]. Their approach involved conversion of dependency relation extracted from the parser into UNL relation and identified pedagogical clues to determine the level of e-content according to the educational taxonomy. They had used Stanford parser, parse tree processing, concept type identification, attribute generation, and rule generation.

English to Tamil machine translation system using UNL had been done by Sridhar *et al.* (2016) [142]. A new sentence formation algorithm was also proposed to rearrange the translated Tamil words to sentences. The translation system was evaluated using BLEU score. A BLEU score of 0.581 was achieved which was an indication that most of the information in the input sentence was retained in the translated sentence.

The development of dictionary entries of Bangla repetition words to integrate them into UNL had been done by Roy *et al.* (2016) [125]. They had outlined the format of dictionary entries of different types of repetition words that came from verb roots and assigned head words and attributes for those words based on the framework of UNL provided by the UNL center.

Agarwal and Kumar (2016) had developed a multilingual cross-domain client application prototype for UNLization and NLization for NLP applications [6]. Additionally, on top of this proposed system, a public platform for developing language-independent applications has been developed and tested by Agarwal and Kumar (2017) [5]. The proposed QA system in this thesis had been integrated with this environment.

Agarwal and Kumar (2018) has developed Analysis module for UNLization of Punjabi text [4]. They had explained the phase wise working of UNLization of Punjabi language with the help of example sentences taken from the corpus provided by UNDL foundation. The F-measure of the system was 0.970, 0.990, and 1.00 for UC-A1, UGO-A1, and AESOP-A1, respectively. UC-A1, UGO-A1, and AESOP-A1 were the corpus provided by UNDL foundation.

Having done an exhaustive survey on various QA systems, it was observed that UNL based QA system will definitely be a major step in NLP to remove language barriers. The proposed UNL based QA system supports multilinguism (all the languages which are part of UNL programme), can be extended to support other foreign languages (which in future might participate in UNL programme) without changing its architecture and code base. Additionally, the system is able to integrate other UNL based NLP applications like machine translation, sentiment analysis, text summarization *etc.*

Chapter Summary

In this chapter, the literature review on the basis of research carried out by researchers in case of QA systems and developments of UNL based resources has been documented. The complete literature review has been divided into two parts, *i.e.*, research activities in question answering system, and research activities in UNL. The comparison of various QA systems has been performed on various parameters like corpus, evaluation metrics used, evaluation metrics values, domain, availability of source code, scalability, online availability multilingual support *etc.*

After doing an exhaustive survey on various QA systems, it has been observed that the proposed UNL based (online available) QA system will definitely be a major step in NLP and removing the language barrier.

Chapter 3

Architecture and Working of Proposed UNL Based QA System

As it has been discussed in the previous chapter of literature review that the existing question answering systems have some major limitations therefore, there is a need of a question answering system which should be web-based and should support multilingualism. Also, the system's source code should be available along with technical/working details so that it could be extended to support other NLP applications. Having such a question answering system will definitely be a major step in NLP and removing the language barrier.

In this chapter, the architecture and working of proposed question answering system have been explained. The proposed question answering system is based on UNL. The brief description about each of the architecture modules and phase wise working of QA system has been explained in this chapter.

The following section of the chapter sets the expectation, understanding, high-level overview of all concepts that would be covered in subsequent chapters of this thesis. It basically lays the foundation of the entire thesis organization.

3.1 Architecture of UNL Based QA System

The architecture of the proposed QA system has been divided into the four modules, *i.e.*, analysis module, UNL crawler, optimizer and generation module as shown in Figure 3.1.

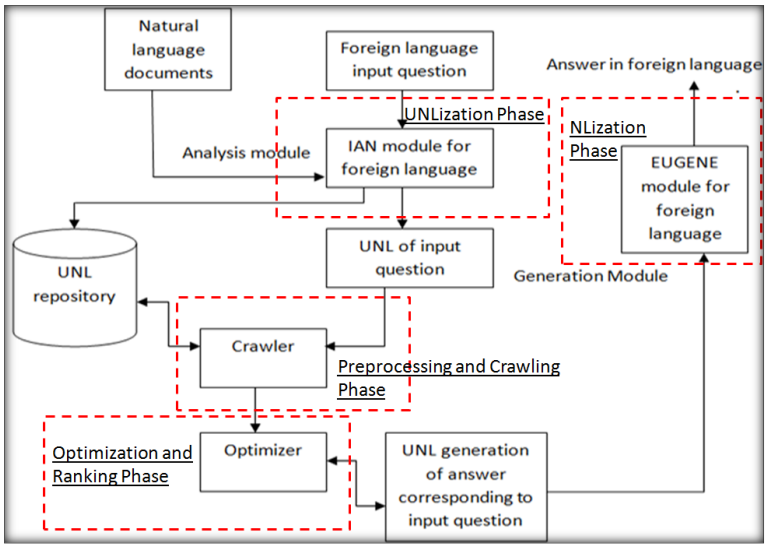


Figure 3.1: Architecture for UNL Based Question Answering System

The analysis module of the proposed question answering system invokes UNLization module of the source natural language to UNLize the question asked by the user and the corpus. The UNL corpus forms the UNL repository. UNL crawler crawls UNL of the question and UNL corpus to find the answer. Optimizer analyses the answer given by UNL crawler and gives ranking to it. This answer after converting to UNL is given as an input to generation module of the proposed question answering system which invokes the NLization module of the target natural language to get the required final answer in that natural language.

The following subsections explain the brief overview of different phases of proposed QA system which includes the purpose and explanation of their corresponding modules. These phases are given as follows.

- UNLization phase,
- Preprocessing and crawling phase,
- Optimizing and ranking phase,
- NLization phase.

3.1.1 UNLization Phase: Analysis Module

The purpose of the UNLization phase is to convert natural language sentence into UNL. In order to UNLize any given corpus or question asked by the user, analysis module uses the UNLization resources, *viz.* natural language dictionary (analysis dictionary), Normalization Rules (NRules), Transformation Rules (TRules) and Disambiguation Rules (DRules) of the source document's natural language. These resources are created according to UNL specifications [153]. The details of UNLization and its resources have been described in Chapter 4 of this thesis. Once these UNLization artifacts have been created for a natural language, then UNLization of the corpus or the question asked by the user can be done by invoking this UNLization/IAN module from the developed interface (by its analysis module) of the proposed question answering system.

The details about the development of UNLization/IAN module along with its sharing and invoking by the analysis module of the proposed system has been explained with the help of example sentences in Chapter 4 of this thesis.

3.1.2 Preprocessing and Crawling Phase: UNL Crawler

UNL crawler is responsible for searching the UNL repository (containing only UNL

of the corpus) depending on the question's UNL generated in the previous step. This module is a part of a developed web-based QA system interface. UNL crawler preprocesses the input UNL and the target UNL repository. It then removes attributes and scope values which are not required for finding the answer. The detailed working of this phase is described in Chapter 5 of this thesis.

3.1.3 Optimizing and Ranking Phase: Optimizer

After finding the probable answers by UNL crawler, optimization phase of question answering system starts. Optimizer analyses and reports if the question asked is correct, answer can be found, and whether the answer provided by UNL crawler is a full match/partial match.

After optimizing phase, in the ranking phase, optimizer gives the rank to the answer found by the UNL crawler. The detailed working and algorithm of this module is described in Chapter 5 of this thesis.

3.1.4 NLization Phase: Generation Module

The NLization phase converts the output of the previous phase to the target natural language by invoking its NLization/EUGENE module. Similar to analysis module, in order to NLize the output, generation module uses generation dictionary, TRules, and DRules created by the computational linguists of the respective natural language and stored at UNL web. These resources are created according to UNL specifications [153]. The detailed working and algorithm of this module is described in Chapter 6 of this thesis.

3.2 Interface and Working of the Proposed QA System

The developed QA system's interface has been divided into two sections as shown in Figure 3.2. The right-hand section (4) of Figure 3.2 is used to share the details of UNLization/IAN and NLization/EUGENE modules. The introduction and working of UNLization/IAN and NLization/EUGENE modules have been explained in Chapter 4 and Chapter 6 respectively of this thesis. The details of how to share and invoke this UNLization/IAN and NLization/EUGENE modules with the help of analysis and generation module respectively have been explained in section 4.11 of Chapter 4 of this thesis.

All the languages whose UNLization/IAN module's details have been shared by the global audience through this interface will appear in the "*Select Source language*"

dropdown as shown in (1) of Figure 3.2. Similarly, all the languages whose NLization/EUGENE module's details have been shared will appear in the "*Select Target language*" dropdown as shown in (2) of Figure 3.2. The third dropdown as shown in (3) of Figure 3.2 is used to select the corpus in which the answer to the question asked will be found. These corpora are stored in the form of UNL as a part of this developed QA system module. The "*Download Corpus*" button allows the user to download the selected corpus.

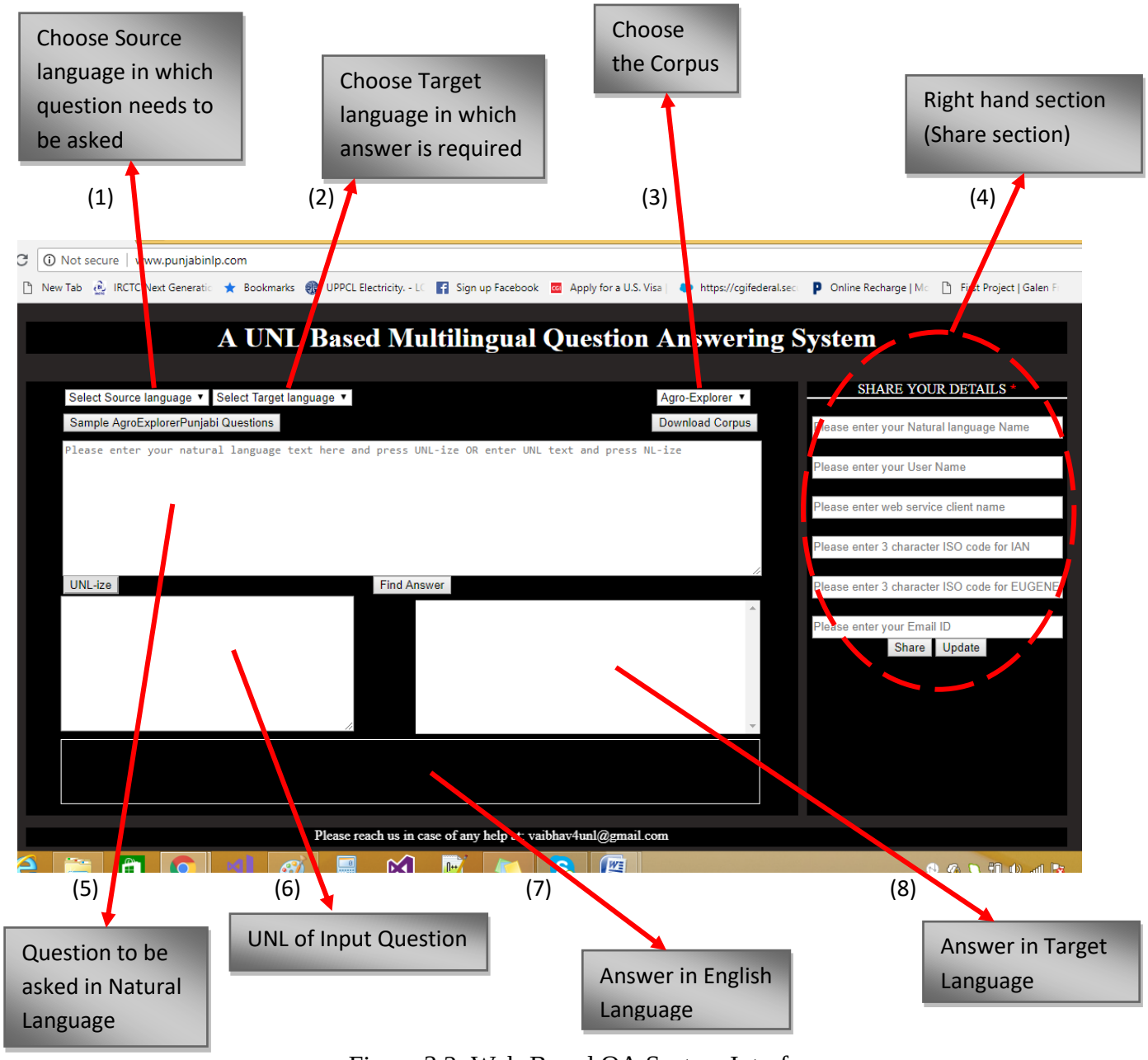


Figure 3.2: Web-Based QA System Interface

After a user selects the corpus, a button appears below the language selection drop-downs which allows the user to select some pre-built questions which are in the same

natural language as selected from the source language drop-down menu. Before asking the question in natural language, first the user needs to select the source language in which the question will be asked. Similarly, the user needs to select the target language from the drop-down in which the user wants the answer. The user can type the question in the selected natural language in the space provided as shown in (5) of Figure 3.2.

After typing the question, when the user clicks the "*Find Answer*" button, analysis module of the source natural language is called and it returns the UNL of the input question which can be seen in the text area as shown in (6) of Figure 3.2. Once UNLization is successful, the UNL crawler and optimizer of developed question answering system works on it and gives the final answer in English keywords in the text area as shown in (7) of Figure 3.2. These English keywords are converted to UNL and to get the answer in target natural language, its generation module is invoked. The final output in target natural language can be seen by the user in the text area as shown in (8) of Figure 3.2.

In the developed question answering system, if the user wants only UNLization of the asked question or any other natural language sentence, then "*UNLize*" button is also provided. On clicking this button, only UNLization will happen and UNL of the natural language sentence will be shown in the text area as shown in (6) of Figure 3.2 but other question answering system's module won't be invoked. This feature can be used by other UNL based natural language processing applications like sentiment analysis system, machine translation system *etc.* which in future can be integrated into this developed system.

Let's say the user selects "*Punjabi*" language from source and target language drop down. Additionally, let's say the user selects "*Agro-Explorer*" corpus. Consider the following example sentence (3.1) taken from Agro-Explorer corpus.

ਆਧੁਨਿਕ ਖੇਤੀਬਾੜੀ ਬਹੁਤ ਹੱਦ ਤੱਕ ਇੰਜਿਨਿਯਰਿੰਗ ਅਤੇ ਤਕਨਾਲੋਜੀ ਅਤੇ ਬਾਇਯੋਲਾਜਿਕਲ ਤੇ ਫਿਜ਼ਿਕਲ ਸਾਇੰਸ ਉੱਪਰ ਨਿਰਭਰ ਹੈ

Ādhunika khētībāṛī bahuta hada taka iñjiniyariṅga atē takanālōjī atē bā'iyōlājikala tē phizikala sā'isa upara nirabhara hai

"Modern agriculture depends heavily on engineering and technology and on the biological and physical science". ...(3.1)

UNL of this example sentence (3.1) is given in (3.2) below and shown with the help of Figure 3.3.

{unl}

aoj(depend.@entry.@present.@pred,agriculture)

scn(depend.@entry.@present.@pred, :03)

man(depend.@entry.@present.@pred, heavily)

and:03(science.@entry.@def.@pl, :01)

mod:03(science.@entry.@def.@pl, :02)

and:02(physical.@entry, biological)

and:01(technology.@entry, engineering)

mod(agriculture, modern)

{/unl} ...(3.2)

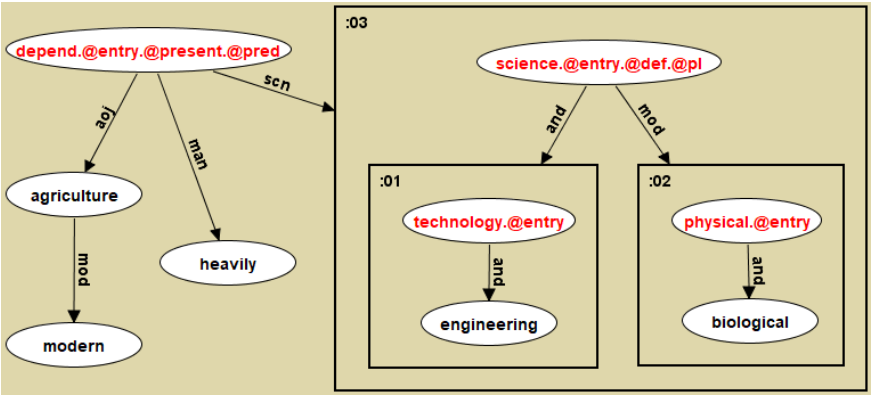


Figure 3.3: UNL Graph of (3.2)

In this UNL, the numbers *":01", ":02",* and *":03"* are called as scopes. Scopes are numbers which represent a relation between nodes and it works as a single semantic entity in a UNL graph. These scope nodes can be used as a node in any other relation of the same UNL to form a hyper-node. For example, in given UNL (3.2) "*and:03(science.@entry.@def.@pl, :01)", "mod:03(science.@entry.@def.@pl, :02)" ,"and:02(physical.@entry, biological)",* and *"and:01(technology.@entry, engineering)*" are hyper-nodes.

Let's assume that user asks the following question given in (3.3) by typing it in the appropriate text area (shown in (5) of Figure 3.2) and clicks on "*Find Answer*" button.

ਆਧੁਨਿਕ ਖੇਤੀਬਾੜੀ ਕਿਹੜੀ ਚੀਜ਼ ਉੱਪਰ ਨਿਰਭਰ ਹੈ?

Ādhunika khētībāṛī kihaṛī cīza upara nirabhara hai?

"Modern agriculture depends on what?" ...(3.3)

After clicking "*Find Answer*" button, analysis module for the Punjabi language is invoked and UNL of this example sentence (3.3) is returned as given in (3.4).

{unl}
aoj(depend@present.@interrogative, agriculture)
scn(depend.@present.@interrogative, 00.01@wh)
mod(agriculture.@interrogative, modern)
{/unl} ...(3.4)

This UNL is further processed by preprocessor and UNL crawler to get following answer in English keywords:

1. *science of physical and biological*
2. *engineering and technology*

The UNL of this answer is then given to generation module of the target language selected from drop down (Punjabi in this case) which outputs the following answer (in the target language):

1. ਸਾਇੰਸ ਆਫ ਫੀਸਿਕਲ ਅਤੇ ਬਾਇਓਲਾਜਿਕਲ
2. ਤਕਨੀਕ ਅਤੇ ਇੰਜਿਨਿਯਰਿੰਗ

3.3 Technology Used to Develop the Proposed QA System

This section provides the technical details like languages, data structures, *etc.*, used to develop the end-to-end QA system. A web-based platform (shown in Figure 3.2) has been developed using HTML, CSS, jQuery, and C#. UNL repository, UNL crawler, optimizer, and the ranking mechanism has been developed and integrated with this platform. The developed system is hosted on www.punjabinlp.com. The data structures used to develop the proposed QA system and details of working of the developed QA system is explained in Section 3.4 of this chapter.

3.4 Data Structures used in the Proposed System

The base data structures, *i.e.*, JSON (JavaScript Object Notation) objects are created for parsing of UNL. JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript programming language, standard ECMA-262 3rd Edition - December 1999. JSON objects are written in key/value pairs.

For storing information of corpus's UNL (after removing attributes and other

irrelevant information), JSON object "*answerObject*" is created as illustrated in Table 3.1. Similarly, for storing information of question's UNL (after removing attributes and other irrelevant information), JSON object "*questionObject*" is created as shown in Table 3.2. These objects have keys as illustrated in Table 3.1 and Table 3.2. These key names are as per lower camel casing coding standard.

For any given UNL (for example UNL as given in 3.2 and 3.4 in this case), index 0 of any key given in Tables 3.1, 3.2, 3.3 represents the corresponding key value of the first relation, index 1 represents the corresponding key value of the second relation, index 3 represents the corresponding key value of third relation and so on. For example, reconsider again the question's UNL given in (3.4) as follows.

{unl}
aoj(depend@present.@interrogative, agriculture)
scn(depend.@present.@interrogative, 00.01@wh)
mod(agriculture.@interrogative, modern)
{/unl}

The value of "*firstArgument*" key of the first relation of this UNL is "*depend*" and can be accessed in code by writing "*firstArgument[0]*". Similarly, the value of "*secondArgument*" key of the first relation is "*agriculture*" and can be accessed by writing "*secondArgument[0]*".

Table 3.1: Data Structure for Storing UNL as Given in (3.4) of Question Information ("*questionObject*" JSON object)

Key	Index	Value	Description
firstArgument	0	depend	Stores first argument of UNL relation
	1	depend	
	2	agriculture	
relationName	0	aoj	Stores name of UNL relation
	1	scn	
	2	mod	
secondArgument	0	agriculture	Stores second argument of UNL relation
	1	00.01	
	2	modern	
otherInfo	0	no	Stores any other information. In the

	1	no	proposed system it has default value ‘no’. This key has been added just to make sure that in future if anything needs to be updated or any other information/ functionality is required, and then this key can be used.
	2	no	

To understand about storage of corpus information, reconsider again the answer UNL given in (3.2).

{unl}

aoj(depend.@entry.@present.@pred,agriculture)

scn(depend.@entry.@present.@pred, :03)

man(depend.@entry.@present.@pred, heavily)

and:03(science.@entry.@def.@pl, :01)

mod:03(science.@entry.@def.@pl, :02)

and:02(physical.@entry, biological)

and:01(technology.@entry, engineering)

mod(agriculture, modern)

{/unl}

The data structure for storing this UNL of corpus information is given and explained in Table 3.2.

Table 3.2: Data Structure for Storing UNL as Given in (3.2) of Corpus Information (“*answerObject*” JSON object)

Key	Index	Value	Description
firstArgument	0	depend	Stores first argument of UNL relation
	1	depend	
	2	depend	
	3	science	
	4	science	
	5	physical	
	6	technology	
	7	agriculture	
	0	aoj	

relationName	1	scn	Stores name of UNL relation
	2	man	
	3	and	
	4	Mod	
	5	And	
	6	And	
	7	Mod	
secondArgument	0	agriculture	Stores second argument of UNL relation
	1	03	
	2	heavily	
	3	01	
	4	02	
	5	biological	
	6	engineering	
	7	modern	
otherInfo	0	no	Stores any other information. In the proposed system it has default value "*no*". This key has been added just to make sure that in future if anything needs to be updated or any other information/ functionality is required, and then this key can be used.
	1	no	
	2	no	
	3	no	
	4	no	
	5	no	
	6	no	
	7	no	

For storing information related to scope nodes, "*hyperNodes*" object is created for corpus's UNL. For the given example sentence (3.1), its "*hyperNodes*" object for storing its UNL (after removing attributes and other irrelevant information) given in (3.4) is as shown in Table 3.3. UNL given in (3.4) has 4 relations with scopes as given in (3.5).

and:03(science.@entry.@def.@pl, :01)
mod:03(science.@entry.@def.@pl, :02)
and:02(physical.@entry, biological)
and:01(technology.@entry, engineering) …(3.5)

In Table 3.3, index value 0 refers to the first relation node of UNL given in (3.5), *i.e.*, "*and*". Similarly, index value 1, 2 and 3 refers to a second, third, and fourth node of UNL given in (3.5), *i.e.*, "*mod*", "*and*", and "*and*" respectively.

Table 3.3: Data Structure for Storing Scope Information ("*hyperNodes*" JSON object) of UNL as given in (3.5)

Key	**Index**	**Value**	**Explanation**
firstArgument	0	science	This key is used to access the first argument of a scope node. For example, "*firstArgument[0] = science*" because in the first relation node of (3.5), *i.e.*, *"and:03(science.@entry .@def.@pl,:01)"* the value of the first argument is "*science*".
	1	science	
	2	physical	
	3	technology	
isScopeOccurenceOnlyOnce	0	0	The value of this key indicates if in a UNL the scope value of a relation node is given to any other relation node or not. 0 indicates false and 1 indicates true. For example, "*isScopeOccurenceOnlyOnce [0] = 0*" because in the first relation node of (3.5), *i.e.*, *"and:03(science.@entry .@def.@pl,:01)"* its scope *i.e.*, 03 appears more than once.
	1	0	
	2	1	
	3	1	
nodeNumber	0	4	The value of this key refers to the position of the scope node in the given UNL. For
	1	5	
	2	6	

	3	7	example, "*nodeNumber[0] = 4*" because the first scope node of (3.5), *i.e.*, *"and:03(science.@entry .@def.@pl,:01)"* has position 4 in (3.2).
numberOfTextArguments	0	1	Few relations in UNL can have a scope as an argument. The value of this key value indicates how many arguments of the scope node have a scope as an argument. 0 indicates that both arguments are scope, 1 indicates that the only argument has a scope, and 2 indicates that both arguments are text *i.e.,* 0 scope values. For example, "*numberOfTextArguments[0] = 1*" because the first relation node of (3.5), *i.e.*, *"and:03(science.@entry .@def.@pl,:01)"* has only one text argument, *i.e.*, "*science*". Its second argument is a scope value.
	1	1	
	2	2	
	3	2	
relName	0	and	The value of this key represents the relation name of the scope node.
	1	mod	
	2	and	
	3	and	
scopeNumber	0	03	The value of this key refers to the value of the scope
	1	03	

	2	02	number. For example, "*scopeNumber[0] = science*" because in the first relation node of (3.5), *i.e.*, *"mod:03(science.@entry .@def .@pl,:02)"* the value of its scope is "*03*".
	3	01	
secondArgument	0	01	This key is used to access the second argument of a scope node. For example "*secondArgument[0] = 02*" because in the first relation node of (3.5), *i.e.*, *"mod:03(science.@entry .@def.@pl,:02)"* the value of the second argument is "*02*".
	1	02	
	2	biological	
	3	engineering	
sentenceNumber	0	1	In the proposed question answering system, the corpus has multiple UNL of all the natural language sentences. So, the value of this key refers to the sentence number to which this particular scope node belongs.
	1	1	
	2	1	
	3	1	

These above mentioned data structures are used in other phases of the developed question answering system and have been discussed in Chapter 5 of this thesis along with working algorithms. The working algorithms, code base, and underlying architecture of the proposed question answering system has been designed and developed in such a way that it need not be changed in order to support foreign languages. However, in order to integrate other UNL based NLP applications in the existing system, the code can be extended without necessarily changing the existing code.

Chapter Summary

In this chapter, the architecture of the proposed question answering system has been described. The requirements and functionalities of all these architecture components have been documented. This chapter also describes the interface of the developed question answering system along with the technology and programming language used.

In this chapter, the base data structures, *i.e.,* JSON objects which are formed during the initial phase of the question answering system have also been highlighted. These data structures are further used by different modules of the developed question answering system to give the final result. This chapter sets the expectation, understanding, high-level overview of all concepts that would be covered in subsequent chapters of this thesis. It basically lays the foundation of the subsequent chapters and thesis organization.

Chapter 4

IAN/UNLization Module for UNLization of Punjabi Language

In the proposed UNL based QAS, user can ask a question in any natural language and can get the output in any natural language. This is possible because the proposed system converts natural language to UNL and works on this generated UNL. Similarly, the output keywords which are in the English language are converted to UNL and are given to the generation module (discussed in Chapter 6) which gives final output in the target natural language. This feature of the proposed question answering system makes it natural language independent. However, in order to UNLize the corpus and question queried by the user, UNLization module (also called as IAN module) of the source language (in which question is being asked) needs to be developed and invoked by the analysis module. UNLization is done with the help of online tool IAN (*i.e., I*nteractive *AN*alyzer) developed by UNDL foundation available at http://dev.undlfoundation.org/analysis/login.jsp. This chapter focuses on the UNLization process and results of the UNLization module for the Punjabi language. This chapter also illustrates how the UNLization module of the source natural language is invoked by the analysis module of the proposed system.

4.1 IAN Framework

IAN requires some artifacts such as input document, dictionary, and grammars in order to UNLize a particular source language. These artifacts can be provided through its interface according to the specifications provided by UNDL. The first artifact is the NL document that is to be UNLized. Second is the NL-UNL (analysis) dictionary. This dictionary is a type of lexical database consisting of mapping of UWs to NL entries accompanied by the corresponding features. The third is the NL-UNL (analysis) transformation grammar. This grammar consists of a number of transformation rules which help in converting natural language sentences into UNL graphs. Fourth is the NL-UNL (analysis) disambiguation grammar. This grammar consists of a number of disambiguation rules which help in improving the output of the tokenization and of the transformation.

IAN tool is a fully automatic tool. It takes natural language document as input and gives an output in form of UNL. IAN has 8 tabs, *i.e.,* welcome, NL input, dictionaries,

NRules, TRules, DRules, IAN console, and compare. The subsequent text gives the details of these tabs.

First is the welcome tab**,** which gives a brief introduction to the tool. Figure 4.1 shows the welcome tab.

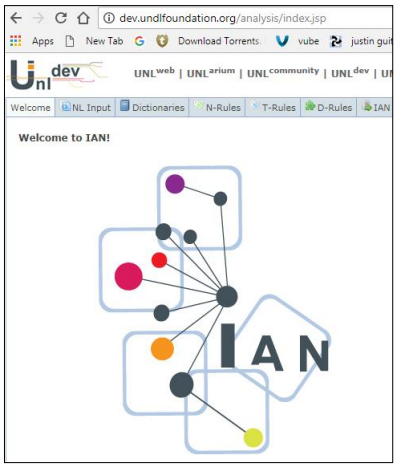


Figure 4.1: Snapshot of the Welcome Tab [68]

Second is the NL input tab, where the natural language document to be UNLized is provided. Figure 4.2 shows the NL input process in the input tab.

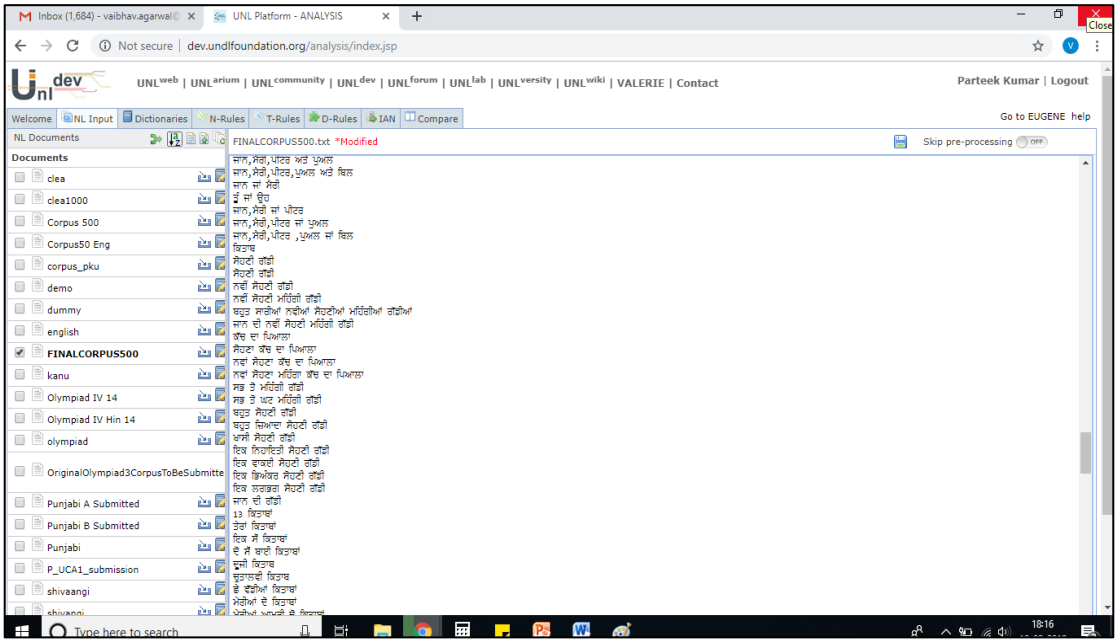


Figure 4.2: Snapshot of NL Input Tab [68]

The third tab is the dictionaries, where NL-UNL dictionaries are provided which are used in natural language analysis. There can be several different dictionaries and can be loaded to process the same corpus. However, it should be kept in mind that order of the dictionaries should be correct while loading because of the order of entries matters

for tokenization in the dictionary. The dictionaries can be reordered using the option "*reorder dictionaries*" available at the top of the menu. Figure 4.3 shows the snapshot of the IAN dictionary for the Punjabi language.

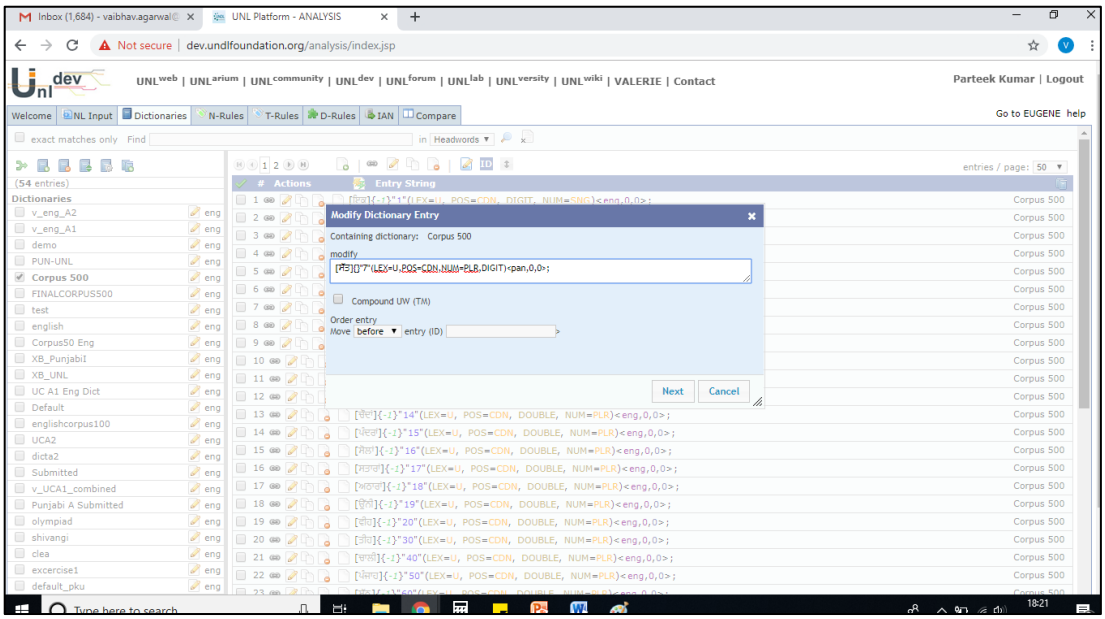


Figure 4.3: Snapshot Showing Creation of Dictionary Under Dictionary Tab [68]

The fourth tab is N-Rules or NRules (*i.e.*, Normalization Rules), where normalization grammar is provided. Figure 4.4 shows the snapshot of N-Rule tab. Normalization has been discussed in section 4.4 of this chapter.

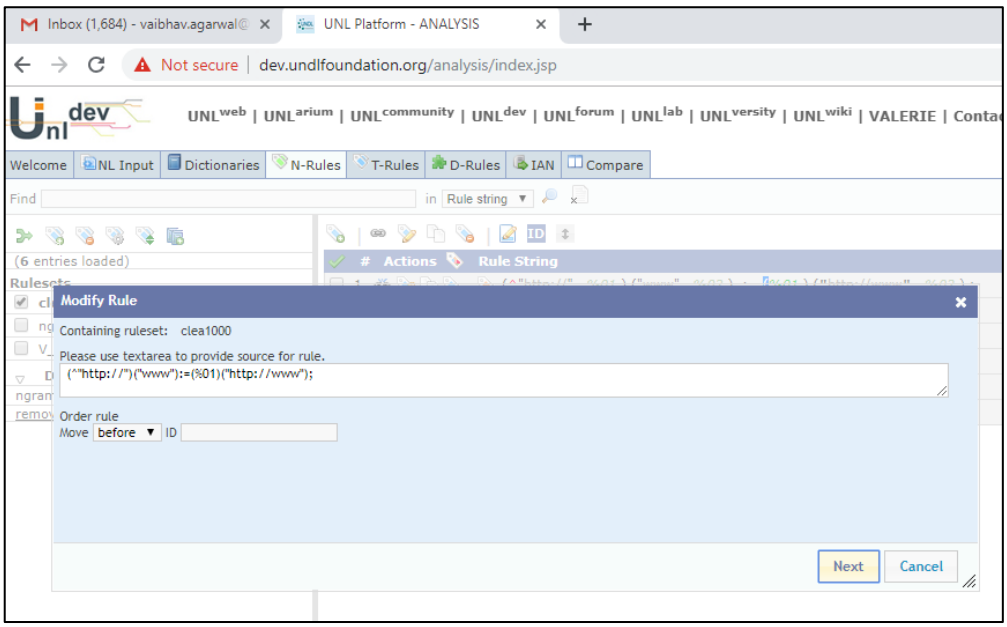


Figure 4.4: Snapshot Showing NRules in NRule Tab [68]

The fifth tab is T-Rules or TRules (*i.e.*, Transformation Rules). In this tab, NL-UNL transformation grammar is provided which is used to process the NL input. Figure 4.5

shows the snapshot of TRule tab.

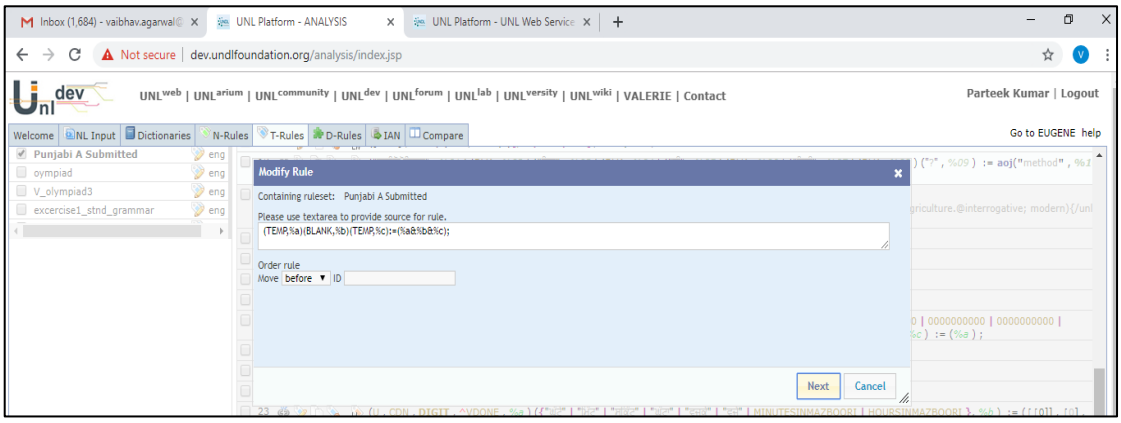


Figure 4.5: Snapshot Showing Creation of TRule in TRule Tab [68]

Sixth is the D-Rules or DRules (*i.e.*, Disambiguation Rules) tab. In this tab, NL-UNL disambiguation grammar is provided which helps in controlling the tokenization and improving the output of the transformation grammar. Figure 4.6 shows the snapshot of D-Rule tab. Disambiguation rules have been explained in section 4.6.

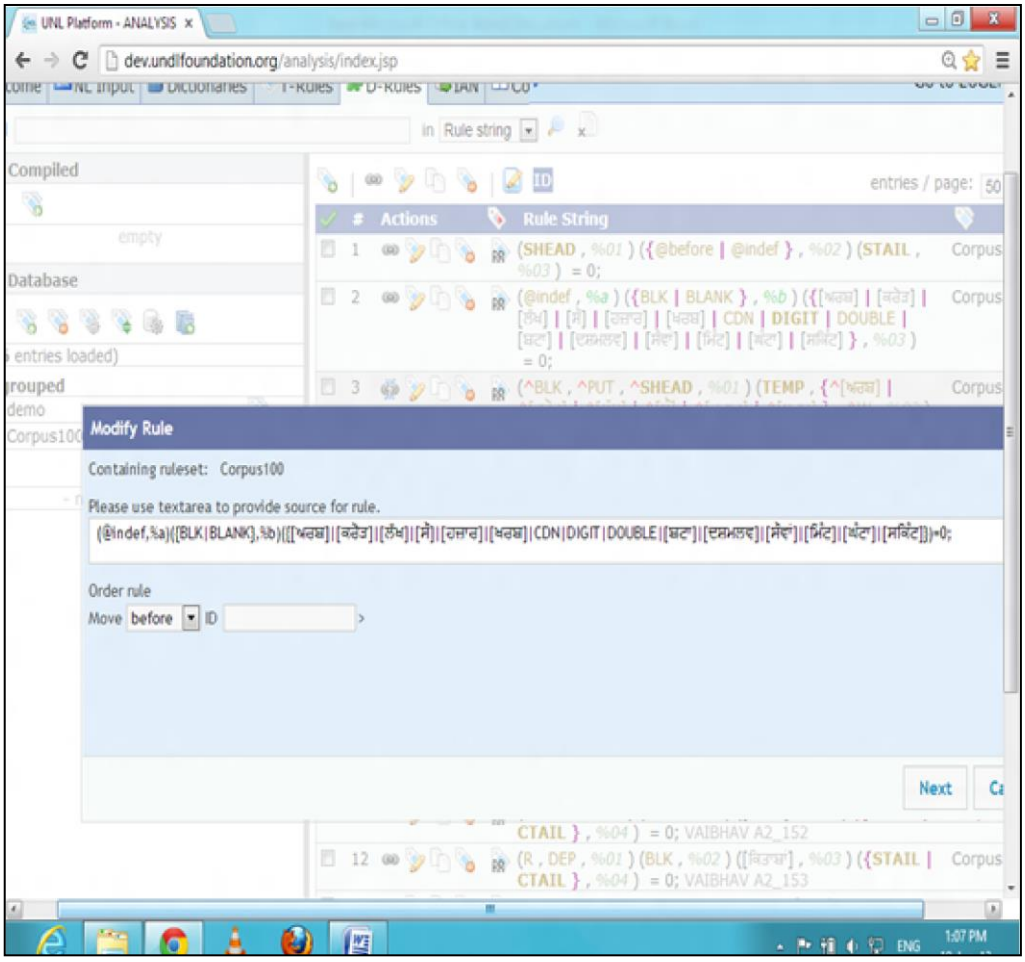


Figure 4.6: Snapshot Showing Creation of DRule Under DRule Tab [68]

The seventh tab is the IAN console tab, where results are seen. A list of sentences in the form of NL input appears in the IAN console tab. These sentences can be processed either one at a time or in a range. IAN console shows the complete UNLization process of the selected natural language sentence. Figure 4.7 shows a snapshot of the IAN console.

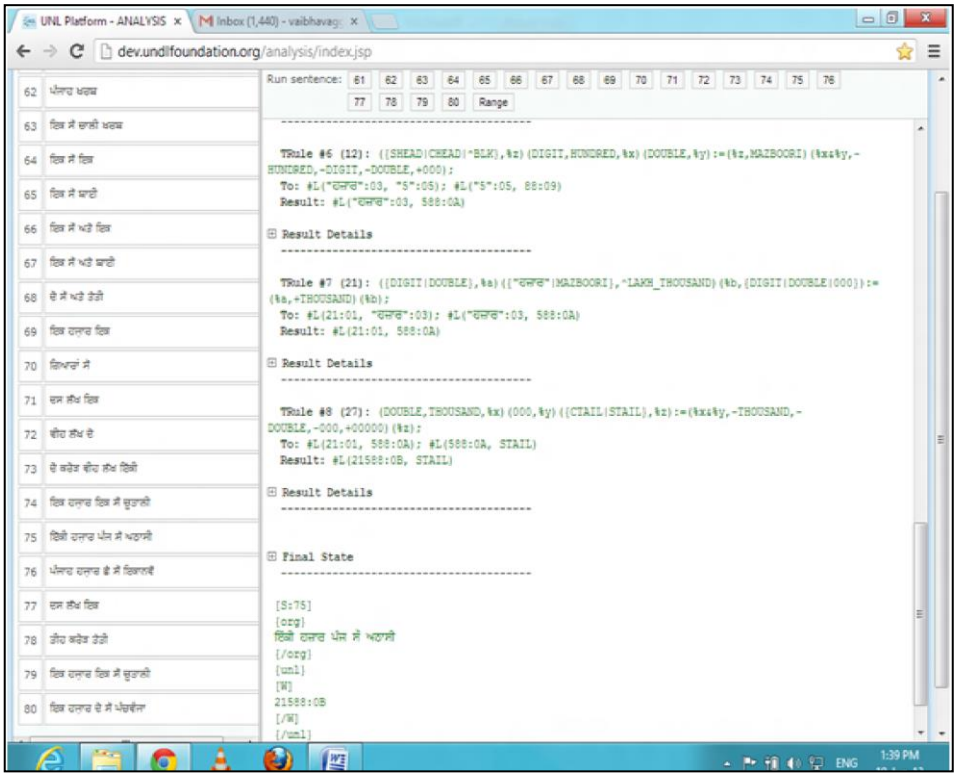


Figure 4.7: Snapshot of IAN Console [68]

The eighth tab is the Compare tab which has the ability to compare the final two results. The details of analysis dictionary and TRules for UNLization have been explained with the help of example sentences in following section.

4.2 Dictionary and Types of Transformation Rules (TRules) for UNLization

Every word present in a corpus to be UNLized should be present in its analysis dictionary or else it will be considered as a temporary word. Once the entries of these words have been created in the analysis dictionary, then after that normalization, tokenization, and disambiguation (as discussed in section 4.4, 4.5, and 4.6), UNLization process starts. UNLization is done with the help of Transformation Rules (TRules). Sections 4.2.1 and 4.2.2 illustrate the Dictionary for UNLization and types of TRules for UNLization.

4.2.1 Dictionary for UNLization

The analysis dictionary that is used in UNLization is a plain text file in which there is a single entry per line. The format of an entry in the dictionary is given in (4.1) [43].

[NLW]{ID}“UW” (ATTR)<FLG,FRE,PRI> ; COMMENTS …(4.1)

Here, “*NLW*” is the Natural Language Word, “*ID*” is the unique identifier (primary-key) of the entry (which is optional), “*UW*” is the corresponding UW, “*ATTR*” represents the list of features of the NLW, “*FLG*” is the three-character language code according to ISO 639-3, “*FRE*” refers to the frequency of NLW and ranges from 0 to 255, “*PRI*” is the priority of the NLW and ranges from 0 to 255. The default values of “*FRE*” and “*PRI*” is 0 and can be updated (optional).

Consider a corpus having an example sentence given in (4.2).

ਖ਼ਰਗੋਸ਼ ਨੇ ਇਕ ਦਿਨ ਕੱਛੂਕੁੰਮੇ ਦੇ ਛੋਟੇ ਪੈਰਾਂ ਅਤੇ ਹੌਲੀ ਚਾਲ ਦਾ ਮਜ਼ਾਕ ਉਡਾਇਆ ...(4.2)

k̲h̲argōsh nē ik din kacchūkummē dē chōṭē pairāṃ atē haulī cāl dā mazāk uḍāiā

“The hare one day ridiculed the short feet and slow pace of the tortoise”.

The analysis dictionary for this corpus must contain twenty two dictionary entries as given in (4.3).

[ਖ਼ਰਗੋਸ਼]{}"hare"(LEX=N,POS=NOU,GEN=MCL ,NUM=SNG)<pan,0,0>;
[ਨੇ]{-1}""(LEX=P,POS=PPS,rel=agt)<pan,0,0>;
[ਇਕ ਦਿਨ]{}"one day"(LEX=N,POS=NOU,rel=tim)<pan,0,0>;
[ਕੱਛੂਕੁੰਮੇ]{}"tortoise"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>;
[ਦੇ]{}""(LEX=P,POS=PPS,rel=mod)<pan,0,0>;
[ਛੋਟੇ]{}"short"(LEX=J,POS=ADJ)<pan,0,0>;
[ਪੈਰਾਂ]{}"foot"(LEX=N,POS=NOU,NUM=PLR)<pan,0,0>;
[ਅਤੇ]{}"and"(LEX=C,POS=COO,rel=and)<pan,0,0>;
[ਹੌਲੀ]{}"slow"(LEX=J,POS=ADJ,NUM=SNG)<pan,0,0>;
[ਚਾਲ]{}"pace"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>;
[ਦਾ]{}""(LEX=D,POS=ART,att=@def)<pan,0,0>;
[ਮਜ਼ਾਕ ਉਡਾਇਆ]{}"ridicule"(LEX=V,POS=VER,att=@past,NUM=SNG)<pan,0,0>;

Eleven blank spaces are also identified as:-

[]{}""(BLK)<pan,0,0>; ...(4.3)

Here, “*LEX”* specifies lexical category, “*N”* specifies noun, “*P”* specifies preposition,

"J" represents adjective, "*C"* represents conjunction, "*D"* represents determiner, "*POS"* represents part-of-speech, "*NOU"* represents common noun, "*PPS"* represents postposition, "*ADJ"* represents adjective, "*COO"* represents coordinating conjunction, "*ART"* represents article, "*VER"* represents full verb, "*GEN"* represents gender, "*MCL"* represents masculine, "*rel"* represents relation, "*agt"* represents agent relation, "*tim"* represents time relation, "*mod"* represents modifier relation, "*and"* represents and relation, "*att"* holds the attribute value of a node, "*@def"* represents definite, "*@past"* represents past attribute, "*NUM"* represents number whose value could be either "*SNG"* for singular or "*PLR"* for plural, and "*BLK"* is the attribute given to the blank space. In <pan,0,0>, pan represents the three-character language code for Punjabi according to ISO 639-3. First "0" refers the frequency of Natural Language Word (NLW) in text and second "0" refers the priority of the NLW.

In case of analysis dictionary, there is an entry for each word appearing in the corpus irrespective of the fact that the word is in its root form or in an inflection of its root word. For example, consider that the corpus containing the example sentence as given in (4.2) above has another sentence as given in (4.4).

ਕੱਛੂਕੁੰਮੇ ਦਾ ਛੋਟਾ ਪੈਰ ...(4.4)

kacchūkummē da chōṭa pair

"Short foot of the tortoise"

Now, although the analysis dictionary of the given corpus for example sentence (4.2) has an entry of Punjabi word "*ਪੈਰਾਂ" "pairāṃ*" *"foot"* as given in (4.3), yet it will have one of the dictionary entries for example sentence (4.4) as "*ਪੈਰ" "pair*" *"foot"* also as given in (4.5).

[ਪੈਰ]{}"foot"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>; …(4.5)

The only difference between these two entries is that the former has "*NUM*" value "*PLR*" whereas latter has this value as "*SNG*".

4.2.2 Types of TRules for UNLization

TRules are used to convert natural language text to UNL as illustrated in section 4.7. In TRules, NL-UNL transformation grammar (*i.e.*, the grammar to be used to process the natural language input) is provided. TRules are classified according to the type of modification that they promote [152].

a) **LL, or list-to-list**, where the initial state and the final state are list structures. There are four different subtypes of LL rules as shown in Table 4.1.

Table 4.1: LL Rules [152]

ACTION	RULE	DESCRIPTION
ADD	(%x):=(%x)(%y);	The node %y is added to the right of the node %x.
	(%x):=(%y)(%x);	The node %y is added to the left of the node %x.
DELETE	(%x):=-(%x);	The node %x is deleted.
	(%x):=;	
REPLACE	(%x):=(%y);	All the instances of node %x will be replaced by node %y.
MERGE	(%x)(%y):=(%x&%y);	The nodes %x and %y will be merged.

b) **TT, or tree-to-tree,** where the initial state and the final state are tree structures. There are three different subtypes of TT rules as shown in Table 4.2.

Table 4.2: TT Rules [152]

ACTION	RULE	DESCRIPTION
ADD RELATION	SYN1(%x;%y):=+SYN2(%w;%z);	SYN is a syntactic relation, and %x, %y, %z, and %w are nodes. The relation SYN2 between the nodes %w and %z will be added to the graph containing the relation SYN1 between the nodes %x and %y.
DELETE RELATION	SYN(%x;%y):=-SYN(%x;%y);	The relation SYN between the nodes %x and %y will be deleted (the nodes %x and %y will not be deleted).
	SYN(%x;%y)=;	
REPLACE RELATION	SYN1(%x;%y):=SYN2(%w;%z);	The relation SYN1 between the nodes %x and %y will be replaced by the relation SYN2 between the nodes %w and %z.

c) **NN, or network-to-network,** where the initial state and the final state are network structures. There are three different subtypes of NN rules as shown in Table 4.3.

Table 4.3: NN Rules [152]

ACTION	RULE	DESCRIPTION
ADD RELATION	SEM1(%x;%y):=+SEM2(%w;%z);	Here SEM is any of the existing UNL relations, and %x, %y, %z, and %w are nodes. The relation SEM2 between the nodes %w and %z will be added to the graph containing the relation SEM1 between the nodes %x and %y.
DELETE RELATION	SEM(%x;%y):=-SEM(%x;%y); SEM(%x;%y)=;	The relation SEM between the nodes %x and %y will be deleted (the nodes %x and %y will not be deleted).
REPLACE RELATION	SEM1(%x;%y):=SEM2(%w;%z);	The relation SEM1 between the nodes %x and %y will be replaced by the relation SEM2 between the nodes %w and %z.

d) **LT, or list-to-tree,** converts lists into trees. There are 2 different subtypes of LT rules as shown in Table 4.4.

Table 4.4: LT Rule [152]

ACTION	RULE	DESCRIPTION
ADD	(%x)(%y):=+SYN(%x;%y);	The relation SYN is created between the nodes %x and %y if there is a linear relation between them (the linear relation is not deleted).
REPLACE	(%x)(%y):=SYN(%x;%y);	The linear relation between %x and %y is replaced by the relation SYN between the same nodes (*i.e.*, the linear relation is deleted).

e) **TN, or tree-to-networks,** convert trees into networks. There are two types of TN rules as shown in Table 4.5.

Table 4.5: TN Rule [152]

ACTION	RULE	DESCRIPTION
ADD	SYN(%x;%y):=+SEM(%w;%x);	The semantic relation SEM between the nodes %w and %x is created if there is a syntactic relation SYN between the nodes %x and %y.
REPLACE	SYN(%x;%y):=SEM(%	The syntactic relation SYN between the nodes %x and %y is replaced by the semantic relation SEM

	x;%y);	between the same nodes.

The working of UNLization/IAN module using IAN (UNLization process) has been divided into different phases. These phases have been discussed along with example sentences in the following sections.

4.3 Phases of UNLization Process

UNLization is a rule-based approach. UNLization process aims to convert NL document to UNL document. UNLization is done using IAN that involves the creation of Normalization Grammar (N-Grammar or NRules), analysis dictionary, Disambiguation Grammar (D-Grammar or DRules), and Transformation Grammar (T-Grammar or TRules) according to the specifications given by UNDL foundation [153]. UNLization works on natural language document. Each of the steps shown in Figure 4.8 has been explained in the following sections.

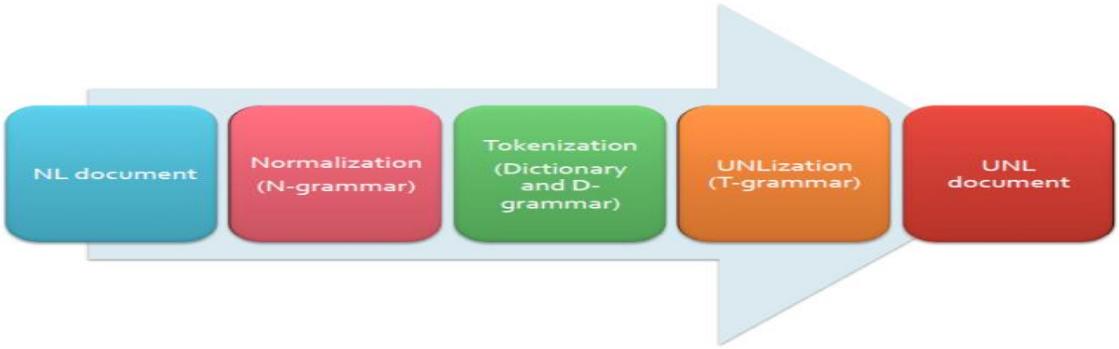


Figure 4.8: UNLization Process Overview

4.4 Normalization

Normalization is the process in which an original sentence given in natural language is converted into more refined form. For example:

a. Replacing contractions

   Don’t > do not, he’ll > he will

b. Replacing abbreviations

   U > you

c. Reordering

   Would you > you would

d. Filling gaps and ellipses

Next week > in the next week

e. Removing extra content

, say, >Ø

Normalization is done with the help of Normalization Grammar (N-Grammar). An example of N-Grammar is given in (4.6).

(%a, [don't]):= (%c,[do not]);
(%b, [dr.]):=(%d,[doctor]); ...(4.6)

Here, "*%a"* refers to node [don't], and "*%b"* refers to node [dr.]. These rules will replace the node "*%a"* with [do not], and the node "*%b"* with [doctor].

Let us consider an example paragraph given in (4.7) as an input text to IAN.

Dr. Peter H. Smith isn't coming on July 1st. He'll be in another meeting in N.Y. I'll check with him another date asap. Would u be available next week, say, around 2 PM? ...(4.7)

The normalized form of example sentence (4.7) is given in (4.8).

Doctor Peter H Smith is not coming on 01/07. He will be in another meeting in New York. I will check with him other date as soon as possible. You would be available next week around 14:00:00? ...(4.8)

4.5 Tokenization

In UNLization, Tokenization refers to splitting the natural language input into nodes, *i.e.*, the tokens or processing units of the UNL framework [69]. During Tokenization, the string like 'hare and tortoise' is split into 5 tokens *viz.* [hare][ ][and][ ][tortoise], if these entries are provided in the dictionary. However, if any word is not found in the dictionary, then that word is considered as a temporary word. An attribute "*TEMP*" is assigned to that word.

The following tokens are created by default by IAN [69].

i. SCOPE – Scope
ii. SHEAD - Sentence head (the beginning of a sentence)
iii. STAIL - Sentence tail (the end of a sentence)
iv. CHEAD - Scope head (the beginning of a scope)
v. CTAIL - Scope tail (the end of a scope)
vi. TEMP - Temporary entry (entry not found in the dictionary)

vii. DIGIT - Any sequence of digits (i.e.: 0,1,2,3,4,5,6,7,8,9)

Tokenization is done by IAN on the basis of following rules.

i. The system first matches the longest entry in the dictionary, from left to right.
ii. The highest frequent entry comes first in case of entries with the same length.
iii. The first to appear in the dictionary comes first in case of entries with the same length and same frequency.
iv. The feature TEMP is assigned to an entry if it is not found in the dictionary.
v. The feature DIGIT is assigned to the strings exclusively formed by digits.
vi. The feature SHEAD (Sentence head) is automatically assigned to the beginning of the paragraph, and the feature STAIL (Sentence tail) is assigned to the end of the paragraph.
vii. No other tokenization and punctuation is done by the system. For example, blank spaces and punctuation signs are not automatically recognized.

4.6 Disambiguation Rules (D-Rules)

Tokenization is also controlled with the help of D-Rules (Disambiguation Grammar). There can be several scenarios where a single natural language word has several dictionary entries. In such cases, D-Rules help in tokenization by selecting the desired dictionary entry. Consider an example sentence given in (4.9).

This is necessary ...(4.9)

Assume that the dictionary entries are as:

[]{}""(BLK)<eng,0,0>;

[this]{}""(LEX = D, POS = DEM, att= @proximal)<eng,0,0>;

[this]{}""(LEX = R, POS = DEP, att= @proximal)<eng,0,0>;

[is]{}""(LEX = I, POS = AUX)<eng,0,0>;

[is]{}""(LEX = V, POS = COP)<eng,0,0>;

[necessary]{}""(LEX = N, POS = NOU)<eng,0,0>;

[necessary]{}""(LEX = J, POS = ADJ)<eng,0,0>;

As seen above "*this*", "*is*", "*necessary*" root words have multiple dictionary entries. Now, "*this*" can be used as determiner or pronoun as indicated by "*LEX = D*", "*LEX=R*" respectively. Similarly, *"is"* could be used as copula verb as indicated by "*LEX = V, POS=COP*" or auxiliary verb as indicated by "*LEX = I, POS = AUX*". The third root word, *i.e.*, "*necessary*" can be used as noun or adjective as referred by

"*LEX=N*" and "*LEX=J*" respectively. Now in order to UNLize the example sentence (4.5), D-Rules will decide that which dictionary entries IAN should consider during tokenization. However, if no D-Rule is specified then by default during tokenization the entry which appeared first in the dictionary list would be chosen. So, for example sentence (4.5), by default tokenization will happen like:

(this,D)(BLK)(is,I)(BLK)(necessary,N)

Although the correct tokenization should happen like:

(this,R)(BLK)(is,I)(BLK)(necessary,N)

Consider the below scenarios in which Tokenization will modify if following D-Grammars are written.

Scenario #1:

i. Let's say the user has written the following D-Rule.

   (D)(BLK)(I):=0;

   According to this rule, the probability of occurrence of a node having "*D*" as a feature, followed by blank space and a node having "*I*" as a feature is zero. Now, because of this D-Rule, once first dictionary entry of "*this*" is selected as a node having "*LEX=D*", dictionary entry of "*is*" would be the second entry having "*LEX=V*" because as per the D-Rule after a node having "*D*" as a feature, followed by blank space, a node having "*I*" as a feature cannot come.

ii. Because of the above D-Rule, **Tokenization** will happen as follows.

   (this,D)(BLK)(is, V)(BLK)(necessary,N)

Scenario #2:

i. Let's say the user has written the following D-Rules.

   (D)(BLK)(I):=0;

   (D)(BLK)(V):=0;

   (V)(BLK)(J):=1;

   According to these rules the probability of occurrence of a node having "*D*" as a feature, followed by blank space, and followed by a node having "*I*" as a feature is zero. Similarly, the probability of occurrence of a node having "*D*" as a feature, followed by blank space, and followed by a node having "*V*" as a feature is zero. The third D-Rule means that the probability of occurrence of a node having "*V*" as a feature, followed by blank space, and followed by a node having "*J*" as a feature is one. Now because of this D-Rule the second dictionary entry of "*this*" is

selected as a node having "*LEX=R*", dictionary entry of "*is*" would be the second entry having "*LEX=V*" and second dictionary entry of "*necessary*" is selected having "*LEX=J*" because as per the D-Rule after a node having "*D*" as a feature, followed by blank space, a node having "*I*" as a feature cannot come and after a node having "*D*" as a feature, followed by blank space, a node having "*V*" as a feature also cannot come but after a node having "*V*" as a feature, followed by blank space, a node having "*J*" as a feature should definitely come .

ii. Because of the above D-Rule, **Tokenization** will happen as follows.
(this,R)(BLK)(is,V)(BLK)(necessary,J)

Scenario #3:

i. Let's say the user has written the following D-Rules.
(D)(BLK)(I):=0;
(D)(BLK)(V):=0;
According to these rules, the probability of occurrence of a node having "*D*" as a feature, followed by blank space, and followed by a node having "*I*" as a feature is zero. Similarly, the probability of occurrence of a node having "*D*" as a feature, followed by blank space, and followed by a node having "*V*" as a feature is also zero. Now because of this D-Rule the second dictionary entry of "*this*" is selected as a node having "*LEX=R*", dictionary entry of "*is*" would be the first entry having "*LEX=I*" and first dictionary entry of "*necessary*" is selected having "*LEX=N*" because as per the D-Rule after a node having "*D*" as a feature, followed by blank space, a node having "*I*" as a feature cannot come and after a node having "*D*" as a feature, followed by blank space, a node having "*V*" as a feature also cannot come.

ii. Because of the above D-Rule, **Tokenization** will happen as follows.
(this,R)(BLK)(is,I)(BLK)(necessary,N)

Now, this is the correct tokenization which should happen for the example sentence given in (4.9). Therefore, D-Rules given in scenario #3 should be created.

4.7 Transformation (UNLization)

After Normalization and Tokenization, UNLization process starts. UNLization is done with the help of Transformation Grammar (T-Rules). Transformation is done by following the X-Bar approach and is explained in section 4.8. A simple

Transformation process is explained by considering an example sentence given in (4.10).

Hare and Tortoise ...(4.10)

After Tokenization of example sentence given in (4.10) with IAN tool, five lexical items are identified as given in (4.11).

[Hare]{}"Hare"(LEX = N, POS = NOU, GEN = MCL)<eng,0,0>;
[]{}""(BLK)<eng,0,0>;
[and]{}""(LEX = C, POS = CCJ, rel = and)<eng,0,0>;
[]{}""(BLK)<eng,0,0>;
[Tortoise]{}"Tortoise"(LEX = N, POS = NOU, GEN = MCL)<eng,0,0>; ...(4.11)

The process of UNLization of example sentence (4.10) is given in Table 4.6.

. Table 4.6: UNLization of Example Sentence (4.10)

Input Sentence: *Hare and Tortoise*		
1.	TRule	(%a,BLK):= ;
	Description	Here, '*%a'* refers to a blank node [] having attribute 'BLK'. This rule is fired twice consecutively and it removes all the blank spaces.
	Action Taken	Original nodes : [Hare][][and]**[]**[Tortoise] Resultant nodes: [Hare][and][Tortoise]
2.	TRule	({SHEAD|CHEAD}, %01) (N , {NOU|PPN}, %a) (C , and , CCJ , %b) (N , {NOU|PPN}, %c) ({STAIL|CTAIL}, %02) := (NA(%c ; %a) , +N , +AND,%d) ;
	Description	Here, '*%01'*, refers to scope head, '*%a*' refers to node [Hare], '*%b*' refers to node [and], '*%c'* refers to node [tortoise], and '*%02*' refers to scope end. This rule resolves a relation '*NA*' whose first and second arguments are '*%c*' and '*%a'* respectively. The new node so formed is given the name '*%d'* and attributes '*AND*' and '*N*' are assigned to this new node.

	Action Taken	Original nodes : [Hare][and][Tortoise] Resultant nodes: NA([Tortoise], [Hare])
3.	TRule	(NA (%a ; %b) , AND , %01):= and(%a;%b);
	Description	Here, '*%a'* refers to node [Tortoise], and '*%b'* refers to node [Hare]. This rule renames '*NA*' relation to actual '*and*' relation. This is the final output generated by IAN.
	Action Taken	Original nodes : NA([Tortoise], [Hare]) Resultant nodes: and([Tortoise], [Hare])

UNL generated by IAN is given in (4.12).

{org}
Hare and Tortoise
{/org}
{unl}
and(tortoise;hare)
{/unl} ...(4.12)

All the three processes namely normalization, tokenization, and transformation are done on each of the sentence in the corpus and then final UNL of the corpus is obtained. The following subsections illustrate UNLization process for all the following major part of speeches.

- Numbers and ordinals
- Temporary words
- Prepositions
- Conjunctions
- Determiners
- Time
- Verbs
- Adjectives

- Nouns, and sentences
- Pronouns

4.7.1 UNLization of Numbers

UNL represents the numbers in figures, written in the form of words. In Punjabi, single-digit and double-digit numbers cannot be generated by TRules unlike English because in Punjabi they have no repeating pattern. For example, in English a double-digit number let's say "*fifty-two*" and "*fifty-nine*" has "fifty" as repeating pattern while entries of "*two*" ,"*nine*" have been made earlier in analysis dictionary. The dictionary entry of every single digit and double digit number like ਇਕ *ik* "*one", ਦੋ dō "two", ਚਾਰ cār "four", ਗਿਆਰਾਂ giārāṃ "eleven", ਪੰਦਰਾਂ pandrāṃ "fifteen" etc.* are made into the dictionary with their respective attributes as shown below in the examples. While using rule based approach, the sequence of rules is very important because the rules are fired in sequence as per matching. For example, TRule 29 will be fired before TRule 34 irrespective of the fact that the nodes matching the criteria of TRule 34 are before the nodes matching the criteria of TRule 29 in the given input string. The UNLization process for the number has been presented using simple example sentence (4.13).

ਤਿੰਨ ਸੌ ਵੀਹ ...(4.13)

tinn sau vīh
three hundred twenty

After the tokenization of example sentence given in (4.13) with IAN tool, five lexical items are identified as shown in (4.14).

[ਤਿੰਨ]{}"3"(LEX=U,POS=CDN,DIGIT, NUM=SNG)<pan,0,0>;
[ਸੌ]{-1}"ਸੌ"(TEMP)<xxx,0,0>;
[ਵੀਹ]{}"20"(LEX=U,POS=CDN,DOUBLE,NUM= PLR)<pan,0,0>;

Two blank spaces are also identified as :-
[]{}" "(BLK)<pan,0,0>; ...(4.14)

Here, "*LEX"* represents lexical category, "*U"* represents numeral, "*POS"* represents part-of-speech, "*CDN"* represents cardinal, "*NUM"* represents number whose value could be either "*SNG"* for singular or "*PLR"* for plural, "*BLK"* is the attribute given to the blank space, "*TEMP"* represents temporary entry, "*DIGIT*" indicates single

digit number like ਇਕ *ik "one"*, ਦੋ *dō "two"*, ਚਾਰ *cār "four" etc.* while "*DOUBLE*" indicates two digit number like ਗਿਆਰਾਂ *giārāṃ "eleven"*, ਪੰਦਰਾਂ *pandrāṃ "fifteen" etc.*

The process of UNLization of example sentence (4.13) has been illustrated in Table 4.7. Here, transliteration of each Punjabi word is shown only in action taken column.

Table 4.7: UNLization Process for Example Sentence (4.13)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK):= ;	Here, *%a* refers to a blank node [] having attribute 'BLK'. This rule is fired twice consecutively and it removes all the blank spaces.	Original nodes : [ਤਿੰਨ][][**ਸੌ**][][ਵੀਹ] *[tinn][][sau][][vīh] [three][][hundred][][twenty]* Resultant nodes: [ਤਿੰਨ][ਸੌ][ਵੀਹ] *[tinn][sau][vīh] [three][hundred] [twenty]*
2.	({DIGIT|DOUBLE}, %a)("ਸੌ")(%b,{DIGIT |DOUBLE}):=(%a, +HUNDRED)(%b);	Here, node [ਸੌ] *[sau] [hundred]* has been deleted and attribute HUNDRED has been given to previous node [ਤਿੰਨ] *[tinn] [three]*.	Original nodes : [ਤਿੰਨ**]**[ਸੌ][ਵੀਹ] *[tinn][sau][vīh] [three][hundred] [twenty]* Resultant nodes : [ਤਿੰਨ][ਵੀਹ] *[tinn][vīh] [three][twenty]*
3.	({SHEAD|CHEAD |^BLK},%a)(DIGIT, HUNDRED,%x) (DOUBLE,%y):=(%a, ATT9)(%x&%y,	Here, SHEAD and CHEAD mean sentence head and scope head, respectively and '^' is used as negation means	Original nodes: **[ਤਿੰਨ][ਵੀਹ]** ***[tinn][vīh]*** ***[three][twenty]*** Resultant nodes:

	-HUNDRED,-DI GIT, -DOUBLE,+000);	logical NOT. This rule concatenates nodes [ਤਿੰਨ] and [ਵੀਹ] to form a single node [ਤਿੰਨਵੀਹ] as shown in the action taken column. All attributes are removed from the final node and attribute '000' is added to it to indicate that it is a three digit number.	[ਤਿੰਨਵੀਹ] *[tinnvīh]* *[threetwenty]* Now, ਤਿੰਨ *ik* '*three*' and ਵੀਹ *vīh* '*twenty*' are replaced by their universal words '3' and '20', respectively, and the final output 320 is generated by IAN as shown in (4.15).

UNL generated by IAN for sentence (4.13) is given in (4.15).

{org}
ਤਿੰਨ ਸੌ ਵੀਹ

{/org}
{unl}
[w] 320:08 [/w]
{/unl} ...(4.15)

Here, ':08' is the scope internally generated by the IAN tool.

UNLization of numbers has further been explained with the help of another example sentence (4.16).

ਪੰਜਾਹ ਖਰਬ ਵੀਹ ਅਰਬ ਤਿੰਨ ਕਰੋੜ ਤੀਹ ਲੱਖ ਅੱਠ ਸੌ ਤਿੰਨ ...(4.16)

pñjāh kharab vīh arab tinn karōṛ tīh lakkh aṭṭh sau tinn

Five trillion twenty billion thirty three million eight hundred three

After tokenization of example sentence given in (4.16) with IAN tool, twenty-one lexical items are identified as given in (4.17).

[ਪੰਜਾਹ]{-1}"50"(LEX=U,POS=CDN,NUM=PLR,DOUBLE)<pan,0,0>;

[ਖਰਬ]{-1}"ਖਰਬ"(TEMP)<xxx,0,0>;

[ਵੀਹ]{}"20"(LEX=U,POS=CDN,DOUBLE,NUM=PLR)<pan,0,0>;

[ਅਰਬ]{-1}" ਅਰਬ"(TEMP)<xxx,0,0>;

[ਤਿੰਨ]{}"3"(LEX=U,POS=CDN,DIGIT,NUM=SNG)<pan,0,0>;

[ਕਰੋੜ]{-1}"ਕਰੋੜ"(TEMP)<xxx,0,0>;

[ਤੀਹ]{}"30"(LEX=U,POS=CDN,DOUBLE,NUM=PLR)<pan,0,0>;

[ਲੱਖ]{-1}"ਲੱਖ"(TEMP)<xxx,0,0>;

[ਅੱਠ]{}"8"(LEX=U,POS=CDN,DIGIT,NUM=SNG)<pan,0,0>;

[ਸੌ]{-1}"ਸੌ"(TEMP)<xxx,0,0>;

Ten blank spaces are also identified as: -

[ ]{}""(BLK)<pan,0,0>; …(4.17)

The process of UNLization of example sentence (4.16) is given in Table 4.8.

Table 4.8: UNLization Process for Example Sentence (4.16)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK):=;	This Rule is fired ten times to delete all the blank nodes as shown in Action taken column.	Original nodes : [ਪੰਜਾਹ][][ਖਰਬ][][ਵੀਹ][] [ਅਰਬ][][ਤਿੰਨ][][ਕਰੋੜ][] [ਤੀਹ][][ਲੱਖ][][ਅੱਠ][][ਸੌ][] [ਤਿੰਨ] *[pñjāh][][kharab][][vīh][][arab][][tinn][][karōr][][tīh][][lakkh][][atth][][sau][][tinn]* *[fifty][][kharab][][twenty][][arab][][three][][crore][][thirty][][lakh][][eight][]*

			[hundred][][three] Resultant nodes : [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][ਸੌ][ਤਿੰਨ] *[pñjāh][kharab][vīh][arab]* *[tinn][karōr][tīh][lakkh]* *[aṭṭh][sau][tinn]* *[fifty][kharab][twenty]* *[arab][three][crore]* *[thirty][lakh][eight]* *[hundred][three]*
2.	({DIGIT |DOUBLE},%a) ("ਸੌ")(%b, {DIGIT |DOUBLE}):= (%a, +HUNDRED) (%b);	Here, *%a* refers to node [ਅੱਠ] having attribute DIGIT and *%b* refers to node [ਤਿੰਨ] having attribute DIGIT. This rule deletes node [ਸੌ] and gives attribute HUNDRED to node *%a*.	Original nodes : [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][ਸੌ][ਤਿੰਨ] *[pñjāh][kharab][vīh][arab]* *[tinn][karōr][tīh][lakkh]* *[aṭṭh][sau][tinn]* *[fifty][kharab][twenty]* *[arab][three][crore]* *[thirty][lakh][eight]* *[hundred][three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ]

			[ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][karōṛ][tīh][lakkh][aṭṭh][tinn]* *[fifty][kharab][twenty][arab][three][crore][thirty][lakh][eight][three]*
3.	({SHEAD|CHEAD|^BLK},%z)({DIGIT|DOUBLE}, HUNDRED,%x)({STAIL|CTAIL|DIGIT},^DOUBLE, ^DOZEN,^00, %y):=(%z)(%x)([[0]],[0],"0", DOZEN,DIGIT)(%y);	Here, *%z* refers to node [ਲੱਖ], *%x* refers to node [ਅੱਠ], and *%y* refers to node [ਤਿੰਨ]. This rule adds a node [0] between *%x* and *%y*. It also attaches the attributes 'DOZEN' and 'DIGIT' to this newly created node.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][karōṛ][tīh][lakkh][aṭṭh][tinn]* *[fifty][kharab][twenty][arab][three][crore][thirty][lakh][eight][three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][0][ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][karōṛ][tīh][lakkh][aṭṭh][0][tinn]*

			[fifty][kharab][twenty] [arab][three][crore] [thirty][lakh][eight][0] [three]
4.	({SHEAD |CHEAD|^BLK |DIGIT},%z) (DIGIT,DOZEN ,%x)(DIGIT,%y):=(%z)(%x& %y,-DOZEN, -DIGIT,+00);	Here, *%z* refers to node [ਅੱਠ], *%x* refers to node [0], and *%y* refers to node [ਤਿੰਨ]. This rule concatenates nodes *%x* and *%y*.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][0][ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh][lakkh] [aṭṭh][0][tinn]* *[fifty][kharab][twenty] [arab][three][crore] [thirty][lakh][eight][0] [three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh][lakkh] [aṭṭh][0tinn] [fifty][kharab][twenty] [arab][three][crore] [thirty][lakh][eight] [0three]*
5.	({SHEAD |CHEAD|^BLK	Here, *%z* refers to node [ਲੱਖ], *%x* refers to node	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ]

	},%z)(DIGIT, HUNDRED,%x)(00,%y):=(%z) (%x&%y,- HUNDRED,- DIGIT,-00, +000);	[ਅੱਠ], and *%y* refers to the node [0ਤਿੰਨ]. This rule concatenates *%x* and *%y* to form the node [ਅੱਠ0ਤਿੰਨ]. The attribute '000' is given to it and existing attributes DIGIT, 00, and HUNDRED are removed.	[ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ][0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh][lakkh] [aṭth][0tinn]* *[fifty][kharab][twenty] [arab][three][crore] [thirty][lakh][eight] [0three]* Resultant nodes : [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh][lakkh] [aṭth0tinn]* *[fifty][kharab][twenty] [arab][three][crore] [thirty][lakh][eight0three]*
6.	({DIGIT |DOUBLE |TRIPLE},%a) ("ਲੱਖ")(%b, {DIGIT |DOUBLE|000 |0000|00000}):= (%a,+LAKH)	Here, node *%a* refers to [ਤੀਹ], *%b* refers to [ਅੱਠ0ਤਿੰਨ]. This rule deletes node [ਲੱਖ] and gives attribute LAKH to node *%a*.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][ਲੱਖ] [ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh][lakkh] [aṭth0tinn]*

	(%b);		*[fifty][kharab][twenty] [arab][three][crore] [thirty][lakh][eight0three]* Resultant nodes : [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ] [ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh] [aṭṭh0tinn]* *[fifty][kharab][twenty] [arab][three][crore] [thirty][eight0three]*
7.	({SHEAD |CHEAD|^BLK },%z)({DIGIT |DOUBLE| TRIPLE}, LAKH,%x)({000|DIGIT |DOUBLE |STAIL|CTAIL} ,^TEMP1, ^00000,%y):= (%z)(%x)([[00]] ,[00],"00", THOUSAND, DOUBLE, LAKH_THOUS	Here, *%z* refers to node [ਕਰੋੜ], *%x* refers to node [ਤੀਹ], and *%y* refers to node [ਅੱਠ0ਤਿੰਨ]. This rule adds a node [00] between *%x* and *%y*. It also attaches the attributes THOUSAND, DOUBLE and LAKH_THOUSAND to this newly created node.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ] [ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh] [aṭṭh0tinn]* *[fifty][kharab][twenty] [arab][three][crore] [thirty][eight0three]* Resultant nodes : [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ]

	AND)(%y);		[ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][00] [ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][karōr][tīh][00][aṭth0tinn]* *[fifty][kharab][twenty][arab][three][crore][thirty][00][eight0three]*
8.	(LAKH_THOUSAND, THOUSAND, %x)(000,%y)({CTAIL|STAIL}, %z):=(%x&%y, -THOUSAND, -000,-DIGIT, -DOUBLE,-LAKH_THOUSAND,+00000)(%z);	Here, *%x* refers to node [00], *%y* refers to node [ਅੱਠ0ਤਿੰਨ], *%z* refers to STAIL. This rule concatenates nodes *%x* and *%y*. Attribute 00000 is given to this node while all other attributes are removed from it.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][00] [ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][karōṛ][tīh][00][aṭth0tinn]* *[fifty][kharab][twenty][arab][three][crore][thirty][00][eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ] [00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][karōr][tīh][00aṭth0tinn]*

			[fifty][kharab][twenty] [arab][three][crore] [thirty][00eight0three]
9.	(DOUBLE, LAKH,%x)(00000,%y)({ CTAIL|STAIL}, %z):=(%x&%y, -LAKH,-DI GIT ,-DOUBLE, -00000, +0000000)(%z);	Here, *%x* refers to node [ਤੀਹ], *%y* refers to node [00ਅੱਠ0ਤਿੰਨ], and *%z* refers to STAIL. This rule concatenates nodes *%x* and *%y* and gives the attribute 0000000 to this concatenated node while removes all other attributes.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ][ਤੀਹ][00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr][tīh] [00aṭṭh0tinn]* *[fifty][kharab][twenty] [arab][three][crore] [thirty][00eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਕਰੋੜ] [ਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab] [tinn][karōr] [tīh00aṭṭh0tinn]* *[fifty][kharab][twenty] [arab][three][crore] [thirty00eight0three]*
10.	({DIGIT |DOUBLE |TRIPLE},%a)	Here, *%a* refers to node [ਤਿੰਨ] , *%b* refers to node	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ]

	("ਕਰੋੜ",%b):= (%a,+CRORE);	[ਕਰੋੜ]. This rule deletes node [ਕਰੋੜ] and gives attribute CRORE to node *%a*.	[ਤਿੰਨ][ਕਰੋੜ] [ਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][karōr]* *[tīh00aṭṭh0tinn]* *[fifty][kharab][twenty][arab][three][crore][thirty00eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][tīh00aṭṭh0tinn]* *[fifty][kharab][twenty][arab][three][thirty00eight0three]*
11.	(DIGIT,CRORE,^ARAB_CRORE,%a)(0000000,%b):=(%a&%b,-0000000, -CRORE, -DIGIT, +00000000);	Here, *%a* refers to node [ਤਿੰਨ] and *%b* refers to node [ਤੀਹ00ਅੱਠ0ਤਿੰਨ]. This rule concatenates nodes *%a* and *%b*. Attribute 00000000 is given to this concatenated node while all other attributes are removed.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨ][ਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab][tinn][tīh00aṭṭh0tinn]* *[fifty][kharab][twenty][arab][three][thirty00eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ]

			[ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab]* *[tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty]* *[arab]* *[threethirty00eight0three]*
12.	({DIGIT |DOUBLE |TRIPLE},%a) ("ਅਰਬ")(%b, {DIGIT |DOUBLE|000 |0000|00000 |000000 |0000000 |00000000 |000000000}):= (%a,+ARAB) (%b);	Here, *%a* refers to node [ਵੀਹ] , and *%b* refers to node [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ]. This rule deletes node [ਅਰਬ] and gives attribute ARAB to node *%a*.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][ਅਰਬ] [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][arab]* *[tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty]* *[arab]* *[threethirty00eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ] [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh]* *[tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty]* *[threethirty00eight0three]*
13.	{DIGIT |DOUBLE |TRIPLE}, ARAB,%x) (00000000,%y): =(%x)([[0]],[0], "0",TEMP3,%z)	Here, *%x* refers to node [ਵੀਹ] , and *%y* refers to node [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ]. This rule adds a node [0] between *%x* and *%y*.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ] [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh]* *[tinntīh00aṭṭh0tinn]*

	(%y);	Attribute TEMP3 is given to this new node.	*[fifty][kharab][twenty] [threethirty00eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][0] [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][0] [tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty][0] [threethirty00eight0three]*
14.	(TEMP3,%a) (00000000,%b): =(%a&%b, TEMP3, -00000000, +000000000);	Here, *%a* refers to node [0], *%b* refers to node [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ]. This rule concatenates nodes *%a* and *%b.* Attribute 000000000 is given to this concatenated node.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ][0] [ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][0] [tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty][0] [threethirty00eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ] [0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh] [0tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty] [0threethirty00eight0three]*
15.	(DOUBLE, ARAB,%a)(	Here, *%a* refers to node node [ਵੀਹ], and *%b*	Original nodes: [ਪੰਜਾਹ][ਖਰਬ][ਵੀਹ]

	000000000,%b): =(%a&%b, -000000000, -ARAB, -DOUBLE, +00000000000);	refers to na ode [0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] .This rule concatenates node *%a* and *%b*. Attribute 00000000000 is given to this concatenated node.	[0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh][0tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty][0threethirty00eight0three]* Resultant nodes: [ਪੰਜਾਹ][ਖਰਬ] [ਵੀਹ0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh0tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty0threethirty00eight0three]*
16.	({DIGIT |DOUBLE |TRIPLE},%a)("ਖਰਬ")(%b, {DIGIT |DOUBLE|000 |0000|00000 |000000 |0000000 |00000000 |000000000 |0000000000 |00000000000}) :=(%a, +KHARAB)	Here , *%a* refers to node [ਪੰਜਾਹ], and *%b* refers to node [ਵੀਹ]. This rule deletes node [ਖਰਬ] and gives attribute KHARAB to node *%a*.	Original nodes: [ਪੰਜਾਹ][ਖਰਬ] [ਵੀਹ0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh][kharab][vīh0tinntīh00aṭṭh0tinn]* *[fifty][kharab][twenty0threethirty00eight0three]* Resultant nodes: [ਪੰਜਾਹ] [ਵੀਹ0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh]*

	(%b);		*[vīh0tinntīh00aṭṭh0tinn]* *[fifty]* *[twenty0threethirty00eight0three]*
17.	(DOUBLE, KHARAB,%a)(00000000000, %b):=(%a&%b, -KHARAB, -DOUBLE, -00000000000, +0000000000000);	Here, *%a* refers to node [ਪੰਜਾਹ], and *%b* refers to node [ਵੀਹ0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ]. This rule concatenates nodes *%a* and *%b*. Attribute 0000000000000 is given to this concatenated node.	Original nodes: [ਪੰਜਾਹ] [ਵੀਹ0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāh]* *[vīh0tinntīh00aṭṭh0tinn]* *[fifty]* *[twenty0threethirty00eight0three]* Resultant nodes: [ਪੰਜਾਹਵੀਹ0ਤਿੰਨਤੀਹ00ਅੱਠ0ਤਿੰਨ] *[pñjāhvīh0tinntīh00aṭṭh0tinn]* *[fiftytwenty0threethirty00eight0three]* Now, ਪੰਜਾਹ *pñjāh* ‘*fifty*’, ਵੀਹ *vīh* ‘*twenty*’, ਤਿੰਨ *tinn* ‘*three*’, ਤੀਹ *tīh* ‘*thirty*’, and ਅੱਠ *aṭṭh* ‘eight’ are replaced by their universal

			words ‘50’, ‘20’, ‘3’, ‘30’ and ‘8’ respectively, and the final output 5020033000803 is generated by IAN as shown in (4.18).

UNL generated by IAN for sentence (4.16) is given in (4.18).

{org}

ਪੰਜਾਹ ਖਰਬ ਵੀਹ ਅਰਬ ਤਿੰਨ ਕਰੋੜ ਤੀਹ ਲੱਖ ਅੱਠ ਸੌ ਤਿੰਨ

{/org}

{unl}

[w] 5020033000803:0F [/w]

{/unl} ...(4.18)

Here ‘:0F’ is the scope internally generated by the IAN tool.

4.7.2 UNLization of Ordinals

Ordinals numbers in Punjabi language contain ਲਾਂ *lāṃ* / ਵਾਂ *vāṃ* / ਥਾ *thā* / ਜਾ *jā* as suffix as in ਪਹਿਲਾਂ *pahilāṃ* “*first*”, ਇੱਕੀਵਾਂ *ikkīvāṃ* “*twenty first*”, ਚੌਥਾ *cauthā* “*fourth*”, ਤੀਜਾ *tījā* “*third*”, respectively. In UNL, ordinals are represented in figures and attribute “*@ordinal*” is given to it to retain its semantics. During UNLization of ordinals, all rules fired will be same, with the only difference that after deleting blank spaces, a new rule as given in (4.19) will be fired. This rule will add “*@ordinal*” attribute to the corresponding node having suffixes ਲਾਂ *lāṃ* / ਵਾਂ *vāṃ* / ਥਾ *thā* / ਜਾ *jā*.

({DIGIT|DOUBLE|CDN},%x)("ਵਾਂ"|"ਲਾਂ"|"ਥਾ"|"ਜਾ"):=(%x,ORD,att=@ordinal) …(4.19)

This rule gives an attribute “*@ordinal*” to node having any one of the attributes “*DIGIT”*, “*DOUBLE”*, or “*CDN”* (being referred as *%x*) preceding any one of the nodes [ਵਾਂ] / [ਲਾਂ] / [ਥਾ] / [ਜਾ]. The UNLization process for ordinals has been presented using simple example sentence (4.20).

ਤਿੰਨ ਸੌ ਵੀਹਵਾਂ ...(4.20)

tinn sau vīhvāṃ

three hundred twentieth

After tokenization of example sentence (4.20) with IAN tool, six lexical items are produced as given in (4.21).

[ਤਿੰਨ]{}"3"(LEX=U,POS=CDN,DIGIT, NUM=SNG)<pan,0,0>;
[ਸੌ]{-1}"ਸੌ"(TEMP)<xxx,0,0>;
[ਵੀਹ]{}"20"(LEX=U,POS=CDN,DOUBLE,NUM= PLR)<pan,0,0>;
[ਵਾਂ]{-1}"ਵਾਂ"(TEMP)<xxx,0,0>;
Two blank spaces are also identified as :-
[]{}" "(BLK)<pan,0,0>; ...(4.21)

The lexical item *"[ਵਾਂ]{-1}"ਵਾਂ"(TEMP)<xxx,0,0>;"* helps in firing of rule given in (4.19) to attach the attribute "*@ordinal*". UNL generated by IAN for example sentence (4.20) is given in (4.22).

{org}
ਤਿੰਨ ਸੌ ਵੀਹਵਾਂ

{/org}
{unl}
[w] 320:09.@ordinal [/w]
{/unl} ...(4.22)

Here, ':09' is the scope internally generated by the IAN tool.

4.7.3 UNLization of Temporary entries

Temporary entries are entries that are not expected to be UNL-ized, such as URL's, e-mail addresses, phone numbers, *etc.* They are not included in the dictionary, because the UNL is the same as the source. Temporary entries are represented, in UNL, between quotes, as natural language entries. For instance, the UNL for "*asfsdfdfdsf*" is "*asfsdfdfdsf*"; the UNL for "*www.undlfoundation.org*" is "*www.undlfoundation.org*"; *etc*. The UNLization process for temporary entries has been presented using simple example sentence (4.23).

Example 1: asfsdfdfdsf asfsdfdfdsf asfsdfdfdsf ...(4.23)

After tokenization of example sentence given in (4.23) with IAN tool, five lexical items are identified, out of which three are the same temporary words and two are the blank spaces, as given in (4.24).

[asfsdfdfdsf]{-1}"asfsdfdfdsf"(TEMP)<xxx,0,0>;
[ ]{}""(BLK)<pan,0,0>; ...(4.24)

The process of UNLization of example sentence (4.23) is given in Table 4.9.

Table 4.9: UNLization Process for Example Sentence (4.23)

S.No	TRule fired	Description	Action Taken
1.	(%a,TEMP) (BLK,%b) (%c,TEMP) :=(%a)(%b, -BLK, +BLANK) (%c) ;	Here, *%a* refers to first temporary node [asfsdfdfdsf] having attribute 'TEMP', *%b* refers to first blank node [], and *%c* refers to second temporary node [asfsdfdfdsf]. This rule is fired twice consecutively to remove attribute 'BLK' from both blank nodes and gives them attribute 'BLANK'. This is done so that this blank node does not get deleted.	Original nodes : [asfsdfdfdsf][][asfsdfdfdsf] [][asfsdfdfdsf] Resultant nodes: [asfsdfdfdsf][][asfsdfdfdsf] [][asfsdfdfdsf]
2.	(%a,TEMP) (BLK,%b) (%c,TEMP) :=(%a&%b &%c);	Here, *%a* refers to first temporary node [asfsdfdfdsf] having attribute 'TEMP', *%b* refers to first blank node [], and *%c* refers to second temporary node [asfsdfdfdsf]. This rule concatenates node *%a*, *%b* and %c as shown in Action Taken column.	Original nodes : [asfsdfdfdsf][][asfsdfdfdsf] [][asfsdfdfdsf] Resultant nodes: [asfsdfdfdsf asfsdfdfdsf] [][asfsdfdfdsf]
3.	(%a,TEMP) (BLK,%b)	Here, *%a* refers to first temporary node [asfsdfdfdsf	Original nodes: [asfsdfdfdsf asfsdfdfdsf]

	(%c,TEMP) :=(%a&%b &%c);	asfsdfdfdsf], *%b* refers to second blank node [], and *%c* refers to third temporary node [asfsdfdfdsf]. This rule concatenates node *%a*, *%b* and %c as shown in Action Taken column.	[][asfsdfdfdsf] Resultant nodes: [asfsdfdfdsf asfsdfdfdsf asfsdfdfdsf] The final output generated by IAN is as shown in (4.25).

4.7.4 UNLization of Prepositions

In UNL, prepositions are represented either by only relations or by both relations and attributes. The UNL of the preposition "*for*" in different context is illustrated in Table 4.10.

Table 4.10: Values of Preposition "*for*"

Value	**UNL**	**Examples**	
		English	**UNL**
Destination	to	flight to Patiala	to(flight, Patiala)
Aim or purpose	pur	prepared for the examination	pur(prepared, examination)
Beneficiary of an action	ben	played for Patiala	ben(played, Patiala)
Object of a desire, intention or perception	obj	thirst for water	obj(thirst, water)
Recipient (direction, addressee) of an action	gol	baked cake for boy	gol(baked, boy)
Amount, extent	man	a fine for ten bahts	man(fine, bahts.@for)
On behalf of, in favor of, in place of	man	cheered for all class	man(cheered, class.@for)
As being	man	misidentified me for the manager	man(misidentified, manager.@for)
Duration	dur	slept for a day	dur(slept, day)

The UNLization process for prepositions has been shown using a simple example sentence (4.26).

ਮੇਜ਼ ਉੱਤੇ ਪੈਰਿਸ ਬਾਰੇ ਤਸਵੀਰਾਂ ਤੋਂ ਬਿਨਾਂ ਕਿਤਾਬ ...(4.26)

mēz uttē pairis bārē tasvīrāṃ tōṃ bināṃ kitāb

The book on the table about Paris without pictures

After tokenization of example sentence (4.26) with IAN tool, thirteen lexical items are identified as given in (4.27).

[ਮੇਜ਼]{}"table"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>;

[ਉੱਤੇ]{}"on"(LEX=P,POS=PRE,rel=plc,att=@on)<pan,0,0>;

[ਪੈਰਿਸ]{}"Paris"(LEX=N,POS=PPN,NUM=SNG,SEM=LCT)<pan,0,0>;

[ਬਾਰੇ]{}"about"(LEX=P,POS=PRE,rel=cnt,att=@about)<pan,0,0>;

[ਤਸਵੀਰਾਂ]{}"picture"(LEX=N,POS=NOU,NUM=PLR)<pan,0,0>;

[ਤੋਂ ਬਿਨਾਂ]{}"without"(LEX=P,POS=PRE,rel=ma n,att=@without)<pan,0,0>;

[ਕਿਤਾਬ]{}"book"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>;

Six blank spaces are also identified as :-

[ ]{}" "(BLK)<pan,0,0>; …(4.27)

The process of UNLization of example sentence (4.26) has been illustrated in Table 4.11.

Table 4.11: UNLization Process for Example Sentence (4.26)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK): =;	Here, *%a* refers to a blank node []. This rule is fired six times and deletes all the blank spaces.	Original nodes : [ਮੇਜ਼][][ਉੱਤੇ][][ਪੈਰਿਸ][] [ਬਾਰੇ][][ਤਸਵੀਰਾਂ][][ਤੋਂ ਬਿਨਾਂ] [][ਕਿਤਾਬ] *[mēz][][uttē][][pairis][] [bārē][][tasvīrāṃ][] [tōṃ binām][][kitāb]* Resultant nodes:

			[ਮੇਜ਼][ਉੱਤੇ][ਪੈਰਿਸ][ਬਾਰੇ] [ਤਸਵੀਰਾਂ][ਤੋਂ ਬਿਨਾਂ] [ਕਿਤਾਬ] *[mēz][uttē][pairis][bārē][tasvīrām][tōm binām][kitāb]*
3.	(N,%a)(P, PRE,rel,att, %b):=(%a,+ att=%b,+rel =%b,+N);		
5.	(N,%a)(P, cnt,%b)(N, %c):=(NA(%c;%a,+att =%b),%d,+ N,+CNT);	Here, *%a* refers to node [ਪੈਰਿਸ], *%b* refers to node [ਬਾਰੇ], and *%c* refers to node [ਤਸਵੀਰਾਂ@without]. This rule resolves a relation 'NA' whose first and second arguments are *%c* and *%a* respectively. The new node so formed is given the name *%d* and attributes 'CNT' and 'N' for same reasons as in previous rule. Second argument of the relation is given attributes of *%b*.	Original nodes: [ਮੇਜ਼@on][ਪੈਰਿਸ][ਬਾਰੇ] [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without])] *[mēz@on][pairis][bārē][NA([kitāb];[tasvīrām@without])]* Resultant nodes: [ਮੇਜ਼@on][NA([NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without])];[ਪੈਰਿਸ @about])] *[mēz@on][NA([NA([kitāb];[tasvīrām@without])];[pairis@about])]*

6.	(N,rel=plc, att,%a)(N, ^rel,%b):=(NA(%b;%a),+PLC,+N) ;	Here, *%a* refers to node [ਮੇਜ਼@on], *%b* refers to node [NA(NA(ਕਿਤਾਬ;ਤਸਵੀਰਾਂ@with out);ਪੈਰਿਸ@about)]. This rule results into a relation 'NA' with first, second arguments as *%b* and *%a* respectively.	Original nodes: [ਮੇਜ਼@on][NA([NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without])];[ਪੈਰਿਸ @about])] *[mēz@on][NA([NA([kitāb] ;[tasvīrāṃ@without])]; [pairis@about])]*
7.	(N,PLR, ^@pl,%a):= (%a,+@pl);	Here, *%a* refers to node [ਤਸਵੀਰਾਂ@without]. This rule adds attribute '@pl' to node *%a*.	Original nodes: [NA([NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without])]; [ਪੈਰਿਸ@about])]; [ਮੇਜ਼@on])] *[NA([NA([NA([kitāb]; [tasvīrāṃ@without])]; [pairis@about])]; [mēz@on])]* Resultant nodes: [NA([NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਪੈਰਿਸ@about])]; [ਮੇਜ਼@on])] *[NA([NA([NA([kitāb]; [tasvīrāṃ@without@pl])]; [pairis @about])]; [mēz@on])]*

8.	(NA(NA(%a;%b), CNT,%w; %c),PLC, %r):=(%w), (NA(%a;%c),+PLC);	Here, *%a* refers to node [NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without@pl])], *%b* refers to node [ਪੈਰਿਸ@about], *%w* refers to [NA([NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without@pl])];[ਪੈਰਿਸ@about])], *%c* refers to node [ਮੇਜ਼@on], *%r* refers to original node as shown in Action Taken column. This rule splits node *%r* into nodes %w and a new node having relation 'NA' with the first and second argument as *%a* and *%c* respectively.	Original nodes: [NA([NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਪੈਰਿਸ@about])]; [ਮੇਜ਼@on])] *[NA([NA([NA([kitāb]; [tasvīrāṃ@without@pl])]; [pairis@about])]; [mēz@on])]* Resultant nodes: Node1: [NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਪੈਰਿਸ@about])] *[NA([NA([kitāb]; [tasvīrāṃ@without@pl])]; [pairis@about])]* Node2: [NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਮੇਜ਼@on])] *[NA([NA([kitāb]; [tasvīrāṃ@without@pl])]; [mēz@on])]*

9.	(NA(NA(%a;%b), MAN,%w; %c),CNT, %r) :=(%w),(NA(%a;%c) ,+CNT);	Here, *%a* refers to node [ਕਿਤਾਬ], *%b* refers to node [ਤਸਵੀਰਾਂ@without@pl], *%c* refers to [ਪੈਰਿਸ@about], *%w* refers to node [NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without@pl])], and *%r* refers to node [NA([NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without@pl])];[ਪੈਰਿਸ@about])]. This rule split node *%r* into nodes *%w* and a new node having relation 'NA' with first and second argument as *%a* and *%c* respectively.	Original nodes: Node1: [NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਪੈਰਿਸ@about])] *[NA([NA([kitāb]; [tasvīrām@without@pl])]; [pairis@ about])]* Node2: [NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਮੇਜ਼@on])] *[NA([NA([kitāb]; [tasvīrām@without@pl])]; [mēz@on])]* Resultant nodes: Node 1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[NA([kitāb]; [tasvīrām@without@pl])]*

			Node 2: [NA([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[NA([kitāb];* *[pairis@about])]* Node3: [NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਮੇਜ਼@on])] *[NA([NA([kitāb];* *[tasvīrām@without@pl])];* *[mēz@on])]*
10.	(NA(NA(%a;%b), MAN,%w; %c),PLC, %r):=(%w), (NA(%a;%c),+PLC);	Here, *%r* refers to node [NA([NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without@pl])]; [ਮੇਜ਼@on])], *%c* refers to node [ਮੇਜ਼@on], *%w* refers to node [NA([ਕਿਤਾਬ];[ਤਸਵੀਰਾਂ @without@pl])], *%a* refers to node [ਕਿਤਾਬ] , *%b* refers to node [ਤਸਵੀਰਾਂ@without@pl].	Original nodes: Node 1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[NA([kitāb];* *[tasvīrām@without@pl])]* Node 2: [NA([ਕਿਤਾਬ];

		This rule split node *%r* into nodes *%w* and a new node having relation ‘NA’ with the first and second argument as *%a* and *%c* respectively. Note that node *%w* is already present and hence redundancy is removed by IAN tool and in final UNL, redundant nodes appear only once. Therefore redundant nodes are shown in Action Taken column only once.	[ਪੈਰਿਸ@about])] *[NA([kitāb]; [pairis@about])]* Node3: [NA([NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]; [ਮੇਜ਼@on])] *[NA([NA([kitāb]; [tasvīrāṃ@without@pl])]; [mēz@on])]* Resultant nodes: Node1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[NA([kitāb]; [tasvīrāṃ@without@pl])]*

			Node2: [NA([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[NA([kitāb]; [pairis@about])]* Node3: [NA([ਕਿਤਾਬ];[ਮੇਜ਼@on])] *[NA([kitāb];[mēz@on])]*
11.	(NA(%a; %b),CNT): =cnt(%a; %b);	Here, *%a* refers to node [ਕਿਤਾਬ], *%b* refers to node [ਪੈਰਿਸ@about]. This rule changes the name of relation from 'NA' to 'cnt' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: Node1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[NA([kitāb]; [tasvīrāṁ@without@pl])]* Node2: [NA([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[NA([kitāb]; [pairis@about])]*

			Node3: [NA([ਕਿਤਾਬ];[ਮੇਜ਼@on])] *[NA([kitāb];[mēz@on])]* Resultant nodes: Node1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[NA([kitāb];* *[tasvīrām@witthout@pl])]* Node2: [cnt([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[cnt([kitāb];* *[pairis@about])]* Node3: [NA([ਕਿਤਾਬ];[ਮੇਜ਼@on])] *[NA([kitāb];[mēz@on])]*
12.	(NA(%a; %b),PLC):= plc(%a;%b)	Here, *%a* refers to node [ਕਿਤਾਬ], *%b* refers to node [ਮੇਜ਼@on]. This rule changes the name of relation from 'NA'	Original nodes: Node1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])]

		to ‘plc’ keeping same arguments as in the original node, as required in the final UNL.	*[NA([kitāb]; [tasvīrāṃ@without@pl])]* Node2: [cnt([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[cnt([kitāb]; [pairis@about])]* Node3: [NA([ਕਿਤਾਬ];[ਮੇਜ਼@on])] *[NA([kitāb];[mēz@on])]* Resultant nodes: Node1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[NA([kitāb]; [tasvīrāṃ@without@pl])]* Node2: [cnt([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[cnt([kitāb]; [pairis@about])]* Node3: [plc([ਕਿਤਾਬ];[ਮੇਜ਼@on])] *[plc([kitāb];[mēz@on])]*
13.	(NA(%a;	Here, *%a* refers to node	Original nodes:

	%b),MAN): =plc(%a; %b)	[ਕਿਤਾਬ], *%b* refers to node [ਤਸਵੀਰਾਂ@without@pl]. This rule changes the name of relation from 'NA' to 'man' keeping same arguments as in the original node, as required in the final UNL.	Node1: [NA([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[NA([kitāb];* *[tasvīrāṃ@without@pl])]* Node2: [cnt([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[cnt([kitāb];* *[pairis@about])]* Node3: [plc([ਕਿਤਾਬ];[ਮੇਜ਼@on])] *[plc([kitāb];[mēz@on])]* Resultant nodes: Node1: [man([ਕਿਤਾਬ]; [ਤਸਵੀਰਾਂ@without@pl])] *[man([kitāb];* *[tasvīrāṃ@without@pl])]*

			Node2: [cnt([ਕਿਤਾਬ]; [ਪੈਰਿਸ@about])] *[cnt([kitāb];* *[pairis@about])]* Node3: [plc([ਕਿਤਾਬ];[ਮੇਜ਼@on])] *[plc([kitāb];[mēz@on])]* Now all the natural language words are replaced by their universal words and final output is generated by IAN.

UNL generated by IAN for sentence (4.26) is specified in (4.28) and its UNL graph is shown in Figure 4.9.

{org}

ਮੇਜ਼ ਉੱਤੇ ਪੈਰਿਸ ਬਾਰੇ ਤਸਵੀਰਾਂ ਤੋਂ ਬਿਨਾਂ ਕਿਤਾਬ

{/org}

{unl}

plc(book, table:01.@on)

cnt(book, paris:05.@about)

man(book, picture:09.@without.@pl)

{/unl} ...(4.28)

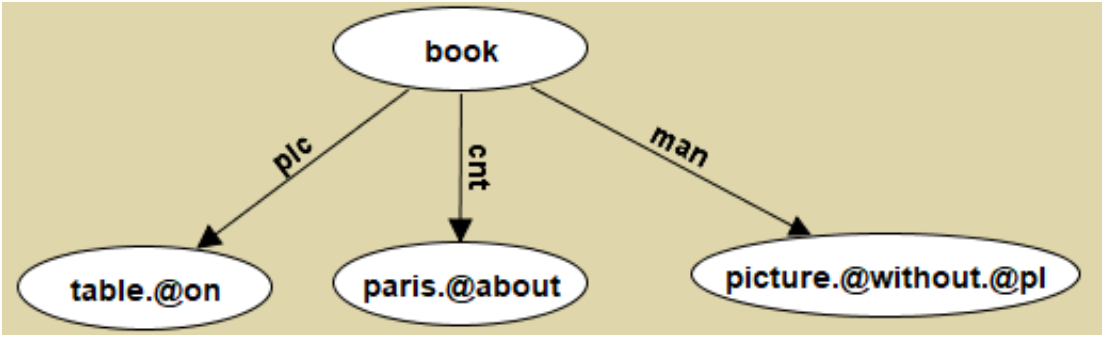


Figure 4.9: UNL Graph of (4.28)

4.7.5 UNLization of Conjunctions

Conjunctions are those lexical items which joins two or more phrases or clauses. In UNL framework, conjunctions are classified as coordinating conjunctions (COO) which joins two or more items of equal syntactic importance ("ਅਤੇ", "ਜਾਂ") and subordinating conjunctions (SCJ), which introduce a dependent clause like "ਕਿ". The UNLization process of conjunctions has been illustrated with the help of a simple example sentence (4.29).

ਜਾਨ, ਮੈਰੀ, ਪੀਟਰ ਅਤੇ ਪਾੱਲ ...(4.29)

jān, mairī, pīṭar atē pāll

John, Mary, Peter and Paul

After the tokenization of example sentence given in (4.29) with IAN tool, ten lexical items are identified as shown in (4.30).

[ਜਾਨ]{}"John"(LEX=N,POS=PPN,GEN=MCL,NUM=SNG,CAS=NOM)<pan,0,0>;

[,]{}""(LEX=C,POS=COO,rel=and)<pan,0,0>;

[ਮੈਰੀ]{}"Mary"(LEX=N,POS=PPN,GEN=FEM,NUM=SNG)<pan,0,0>;

[ਪੀਟਰ]{}"Peter"(LEX=N,POS=PPN,GEN=MCL,NUM=SNG,CAS=NOM)<pan,0,0>

[ਅਤੇ]{}"and"(LEX=C,POS=COO,rel=and)<pan,0,0>;

[ਪਾੱਲ]{}"Paul"(LEX=N,POS=PPN,NUM=SNG)<pan,0,0>;

Four blank spaces are also identified as: -

[ ]{}""(BLK)<pan,0,0>; ...(4.30)

Here, "*GEN=MCL*" means that gender is masculine, "*CAS=NOM"* indicates that case is nominal, "*POS=COO"* indicates that node's part of speech is coordinating conjunction.The process of UNLization of example sentence (4.29) has been illustrated in Table 4.12.

Table 4.12: UNLization Process for Example Sentence (4.29)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK):	Here, *%a* refers to a blank	Original nodes :

	=;	node []. This rule is fired four times and deletes all the blank spaces.	[ਜਾਨ][,][][ਮੈਰੀ][,][][ਪੀਟਰ] [][ਅਤੇ][][ਪਾੱਲ] *[jān][,][][mairī][,][]* *[pītar][][atē][][pāll]* *[John][,][][Mary][,][]* *[Peter][][and][][Paul]* Resultant nodes: [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ] [ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar]* *[atē][pāll]* *[John][,][Mary][,]* *[Peter][and][Paul]*
2.	(N,^COMMAAND,%a)(C,COO,and,%b)(N,^COMMAAND,%c):=(%a,COMAAND)(%b)(%c,COMMAAND);	Here, *%a* refers to node [ਪੀਟਰ], *%b* refers to node [ਅਤੇ], *%c* refers to [ਪਾੱਲ]. This rule gives attribute ‘COMMAAND’ to nodes *%a* and *%c*.	Original nodes : [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ] [ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar]* *[atē][pāll][John][,][Mary]* *[,] [Peter][and][Paul]* Resultant nodes: [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ] [ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar]* *[atē][pāll]*

			[John][,][Mary][,][Peter] [and][Paul]
3.	(N,^COMMAAND,%a) (C,COO,%b)(N,COMMAAND,%c) :=(%a ,+COMMAAND) (%b)(%c);	Here, *%a* refers to node [ਮੈਰੀ], *%b* refers to node [,], *%c* refers to [ਪੀਟਰ]. This rule gives attribute 'COMMAAND' to node *%a*.	Original nodes: [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ] [ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar] [atē][pāll]* *[John][,][Mary][,][Peter] [and][Paul]* Resultant nodes: [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ] [ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar] [atē][pāll]* *[John][,][Mary][,][Peter] [and][Paul]*
4.	(N,^COMMAAND,%a) (C,COO,%b)(N,COMMAAND,%c) :=(%a, +COMMAAND) (%b)(%c);	Here, *%a* refers to node [ਜਾਨ], *%b* refers to node [,], *%c* refers to [ਮੈਰੀ]. This rule gives attribute 'COMMAAND' to node *%a*.	Original nodes: [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ] [ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar] [atē][pāll]* *[John][,][Mary][,][Peter] [and][Paul]* Resultant nodes: [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ]

			[ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar][atē][pāll]* *[John][,][Mary][,][Peter][and][Paul]*
5.	(N,PPN,COMMAAND,%a)(C,and,COO,%b)(N,PPN,COMMAAND,%c):=(NA(%c;%a),+N,+PPN,AND,COMMAAND);	Here, *%a* refers to node [ਜਾਨ], *%b* refers to node [,], and *%c* refers to node [ਮੈਰੀ]. This rule resolves a relation 'NA' whose first and second arguments are *%c* and *%a* respectively. The new node so formed is given attributes 'N', 'PPN', 'AND' and 'COMMAAND'.	Original nodes: [ਜਾਨ][,][ਮੈਰੀ][,][ਪੀਟਰ] [ਅਤੇ][ਪਾੱਲ] *[jān][,][mairī][,][pītar][atē][pāll]* *[John][,][Mary][,][Peter][and][Paul]* Resultant nodes: [NA([ਮੈਰੀ];[ਜਾਨ])][,] [ਪੀਟ ਰ][ਅਤੇ][ਪਾੱਲ] *[NA([mairī];[jān])][,][pīta r][atē][pāll]* *[NA([Mary];[John])][,][Peter][and][Paul]*
6.	(N,PPN,COMMAAND,%a)(C,and,COO,%b)(N,PPN,COMMAAND,%c):=(NA(	Here, *%a* refers to node [NA([ਮੈਰੀ]; [ਜਾਨ])], *%b* refers to node [,] and *%c* refers to node [ਪੀਟਰ]. This rule resolves a relation 'NA' whose first and second arguments are *%c* and	Original nodes: [NA([ਮੈਰੀ];[ਜਾਨ])][,] [ਪੀਟਰ][ਅਤੇ][ਪਾੱਲ] *[NA([mairī];[jān])][,][pīta r][atē][pāll]*

	%c;%a),+N,+PPN,AND,COMMAAND);	*%a* respectively. The new node so formed is given attributes 'N', 'PPN', 'AND' and 'COMMAAND'.	*[NA([Mary];[John])][,][Peter][and][Paul]* Resultant nodes: [NA([ਪੀਟਰ];[NA([ਮੈਰੀ]; [ਜਾਨ])])][ਅਤੇ][ਪਾੱਲ] *[NA([pīṭar];[NA([mairī];[jān])])] [atē][pāll]* *[NA([Peter];[NA([Mary];[John])])] [and][Paul]*
7.	({SHEAD|CHEAD})(N,{NOU|PPN},%a)(C,and,CCJ,%b)(N,{NOU|PPN},%c)({STAIL|CTAIL}):=(NA(%c;%a),+N,+AND);	Here, *%a* refers to node [NA([ਪੀਟਰ];[NA([ਮੈਰੀ];[ਜਾਨ])])], *%b* refers to node [ਅਤੇ], and *%c* refers to node [ਪਾੱਲ]. This rule resolves a relation 'NA' whose first and second arguments are *%c* and *%a* respectively. The new node so formed is given attributes 'N', and 'AND'.	Original nodes: [NA([ਪੀਟਰ];[NA([ਮੈਰੀ]; [ਜਾਨ])])][ਅਤੇ][ਪਾੱਲ] *[NA([pīṭar];[NA([mairī];[jān])])][atē][pāll]* *[NA([Peter];[NA([Mary];[John])])][and][Paul]* Resultant nodes: [NA([ਪਾੱਲ];[NA([ਪੀਟਰ]; [NA([ਮੈਰੀ];[ਜਾਨ])])])] *[NA([pāll];[NA([pīṭar];[NA([mairī];[jān])])])]* *[NA([Paul];[NA([Peter];[NA([Mary];[John])])])]*
8.	(NA(%a;	Here, *%a* refers to node [ਪਾੱਲ],	Original nodes:

	%b),AND): =and(%a; %b);	*%b* refers to node [NA([ਪੀਟਰ];[NA([ਮੈਰੀ];[ਜਾਨ])])]. This rule changes the name of relation from 'NA' to 'and' keeping same arguments as in the original node, as required in the final UNL.	[NA([ਪਾੱਲ];[NA([ਪੀਟਰ]; [NA([ਮੈਰੀ];[ਜਾਨ])])])] *[NA([pāll];[NA([pīṭar];* *[NA([mairī];[jān])])])]* *[NA([Paul];[NA([Peter];* *[NA([Mary];[John])])])]* Resultant nodes: [and([ਪਾੱਲ];[NA([ਪੀਟਰ]; [NA([ਮੈਰੀ];[ਜਾਨ])])])] *[and([pāll];[NA([pīṭar];* *[NA([mairī];[jān])])])]* *[and([Paul];[NA([Peter];* *[NA([Mary];[John])])])]*
9.	(NA(%a; %b),AND): =and(%a; %b);	Here, *%a* refers to node [ਪੀਟਰ], *%b* refers to node [NA([ਮੈਰੀ];[ਜਾਨ])]. This rule changes the name of relation from 'NA' to 'and' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: [and([ਪਾੱਲ];[NA([ਪੀਟਰ]; [NA([ਮੈਰੀ];[ਜਾਨ])])])] *[and([pāll];[NA([pīṭar];* *[NA([mairī];[jān])])])]* *[and([Paul];[NA([Peter];* *[NA([Mary];[John])])])]* Resultant nodes: [and([ਪਾੱਲ];[and([ਪੀਟਰ];

			[NA([ਮੈਰੀ];[ਜਾਨ])])])] *[and([pāll];[and([pīṭar];* *[NA([mairī];[jān])])])]* *[and([Paul];[and([Peter];* *[NA([Mary];[John])])])]*
10.	(NA(%a; %b),AND): =and(%a; %b);	Here, *%a* refers to node [ਮੈਰੀ], *%b* refers to node [ਜਾਨ]. This rule changes the name of relation from 'NA' to 'and' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: [and([ਪਾੱਲ];[and([ਪੀਟਰ]; [NA([ਮੈਰੀ];[ਜਾਨ])])])] *[and([pāll];[and([pīṭar];* *[NA([mairī];[jān])])])]* *[and([Paul];[and([Peter];* *[NA([Mary];[John])])])]* Resultant nodes: [and([ਪਾੱਲ];[and([ਪੀਟਰ]; [and([ਮੈਰੀ];[ਜਾਨ])])])] *[and([pāll];[and([pīṭar]* *;[and([mairī];[jān])])])]* *[and([Paul];[and([Peter];* *[and([Mary];[John])])])]* Now all the natural language words are replaced by their universal words. Also, the inner relations are replaced by their scopes which are

			generated internally by IAN. Note that, here internal relations are not resolved. The final output is generated by IAN as shown in (4.31).

UNL generated by IAN for sentence (4.29) is specified in (4.31) and its UNL graph is shown in Figure 4.10.

{org}
ਜਾਨ, ਮੈਰੀ, ਪੀਟਰ ਅਤੇ ਪਾੱਲ

{/org}
{unl}
and(Paul,:04)
and:04(Peter, :05)
and:05(Mary, John)
{/unl} ...(4.31)

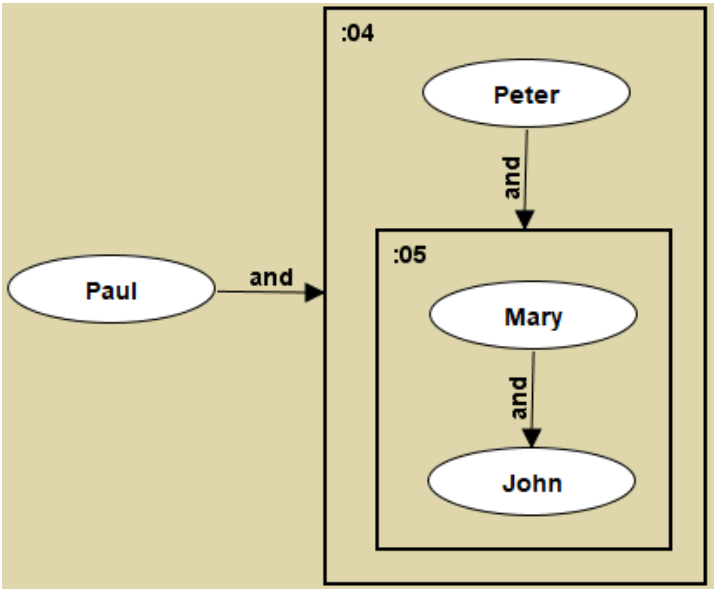


Figure 4.10: UNL Graph of (4.31)

Here, ‘:04’ and ‘:05’ are the scopes internally generated by the IAN tool.

4.7.6 UNLization of Determiners

Determiners are noun-modifiers which express the reference of a noun or noun-phrase in the context, including quantity. For example "*my*", "*your*", "*this*", "*that*" *etc*. The UNLization process for ordinals has been presented using a simple example sentence (4.32).

ਉਹਦੀਆਂ ਬਹੁਤ ਸਾਰੀਆਂ ਕਿਤਾਬਾਂ ...(4.32)

uhdīāṃ bahut sārīāṃ kitābāṃ

His many books

After the tokenization of example sentence given in (4.32) with IAN tool, five lexical items are identified as shown in (4.33).

[ਉਹਦੀਆਂ]{}"00.@3.@male"(LEX=D,POS=POD)<pan,0,0>;

[ਬਹੁਤ ਸਾਰੀਆਂ]{}"many"(LEX=D,POS=QUA,att =@multal)<pan,0,0>;

[ਕਿਤਾਬਾਂ]{}"book"(LEX=N,POS=NOU,NUM=PLR)<pan,0,0>;

Two blank spaces are also identified as:-

[]{}""(BLK)<pan,0,0>; ...(4.33)

Here, "*LEX=D"* implies that node's lexical category is a determiner, "*POS=QUA*" implies that part-of-speech is quantity, "*POS=POD*" implies that part-of-speech is a possessive determiner. The UW "*many*" is associated with an attribute "*@multal*". The process of UNLization of example sentence (4.32) has been illustrated in Table 4.13.

Table 4.13: UNLization Process for Example Sentence (4.32)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK): =;	Here, *%a* refers to blank node []. This rule is fired twice and deletes all the blank spaces.	Original nodes : [ਉਹਦੀਆਂ][][ਬਹੁਤ ਸਾਰੀਆਂ][] [ਕਿਤਾਬਾਂ] *[uhdīām][][bahut sārīām][] [kitābāṃ]* *[His][][many][][books]*

			Resultant nodes: [ਉਹਦੀਆਂ][ਬਹੁਤ ਸਾਰੀਆਂ] [ਕਿਤਾਬਾਂ] *[uhdīām][bahut sārīām]* *[kit ābām]* *[His][many][books]*
2.	(D,att,%a)(N,%b):=(%b,+att=%a);	Here, *%a* refers to node [ਬਹੁਤ ਸਾਰੀਆਂ], *%b* refers to node [ਕਿਤਾਬਾਂ]. This rule deletes the node *%a* and gives its attributes to node *%b*.	Original nodes : [ਉਹਦੀਆਂ][ਬਹੁਤ ਸਾਰੀਆਂ] [ਕਿਤਾਬਾਂ] *[uhdīām][bahut sārīām]* *[kit ābām]* *[His][many][books]* Resultant nodes: [ਉਹਦੀਆਂ][ਕਿਤਾਬਾਂ@multal] *[uhdīāṃ][kitābāṃ@multal]* *[His][books@multal]*
3.	(N,PLR,^@pl,%a):=(%a,+@pl);	Here, *%a* refers to node [ਕਿਤਾਬਾਂ]. This rule gives attribute '@pl' to this node.	Original nodes: [ਉਹਦੀਆਂ][ਕਿਤਾਬਾਂ@multal] *[uhdīāṃ][kitābāṃ@multal]* *[His][books@multal]* Resultant nodes: [ਉਹਦੀਆਂ] [ਕਿਤਾਬਾਂ@multal@pl] *[uhdīām]* *[kitābām@multal@pl]* *[His][books@multal@pl]*

4.	(D,POD,%a)(%b):=(NA(%b;%a),+POS,%d);	Here, *%a* refers to node [ਉਹਦੀਆਂ], *%b* refers to node [ਕਿਤਾਬਾਂ@multal@pl]. This rule resolves a relation 'NA' whose first and second argument are *%b* and *%a* respectively. This new node so formed is given an attribute 'POS'.	Original nodes: [ਉਹਦੀਆਂ] [ਕਿਤਾਬਾਂ@multal@pl] *[uhdīām]* *[kitābām@multal@pl]* *[His][books@multal@pl]* Resultant nodes: [NA([ਕਿਤਾਬਾਂ@multal@pl]; [ਉਹਦੀਆਂ])] *[NA([kitābāṃ@multal@pl];* *[uhdīāṃ])]* *[NA([books@multal@pl]* *;[uhdīāṃ])]*
5.	(NA(%a;%b),POS):=pos(%a;%b);	Here, *%a* refers to node [ਕਿਤਾਬਾਂ@multal@pl], and *%b* refers to node [ਉਹਦੀਆਂ]. This rule changes the name of relation from 'NA' to 'pos' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: [NA([ਕਿਤਾਬਾਂ@multal@pl]; [ਉਹਦੀਆਂ])] *[NA([kitābām@multal@pl];* *[uhdīāṃ])]* *[NA([books@multal@pl];* *[uhdīāṃ])]* Resultant nodes: [pos([ਕਿਤਾਬਾਂ@multal@pl]; [ਉਹਦੀਆਂ])] *[pos([kitābām@multal@pl]* *;[uhdīāṃ])]* Now all the natural language words are replaced

			by their universal words and final output is generated by IAN as shown in (4.34).

UNL generated by IAN for sentence (4.32) is given in (4.34).

{org}

ਉਹਦੀਆਂ ਬਹੁਤ ਸਾਰੀਆਂ ਕਿਤਾਬਾਂ

{/org}

{unl}

pos(book:05.@multal.@pl, 0:01.@3.@male)

{/unl} …(4.34)

Here, ‘:01’, ‘:05’ are the scopes internally generated by the IAN tool.

4.7.7 UNLization of Time

UNLization of time is almost similar as that of numbers because like numbers, their UNL representation is realistic, *i.e.*, without the use of any relation or attribute in final output. The UNLization process for time has been presented using a simple example sentence (4.35).

ਪੰਦਰਾ ਮਿੰਟ ...(4.35)

Pandrā miṇṭ

fifteen minutes

After the tokenization of example sentence given in (4.35) with IAN tool, three lexical items are identified as shown in (4.36).

[ਪੰਦਰਾ]{}"15"(LEX=U,POS=CDN,DOUBLE)<pan,0,0>;

[]{}""(BLK)<pan,0,0>;

[ਮਿੰਟ]{-1}"ਮਿੰਟ"(TEMP)<xxx,0,0>; ...(4.36)

The process of UNLization of example sentence (4.35) has been illustrated in Table 4.14.

Table 4.14: UNLization Process for Example Sentence (4.35)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK):	Here, *%a* refers to blank node	Original nodes :

	=;	[]. This rule deletes the blank spaces.	[ਪੰਦਰਾ][][ਮਿੰਟ] *[Pandrā][][miṇṭ]* *[Fifteen][][minutes]* Resultant nodes: [ਪੰਦਰਾ][ਮਿੰਟ] *[Pandrā][miṇṭ]* *[Fifteen][minutes]*
2.	(U,CDN,{ MYTIME| DOUBLE}, %a)({"ਮਿੰਟ" |MINUTES MZ},%b):= (%a, MINUTES) ;	Here, *%a* refers to node [ਪੰਦਰਾ], *%b* refers to node [ਮਿੰਟ]. This rule deletes the node *%b* and gives an attribute 'MINUTES' to node *%a*.	Original nodes : [ਪੰਦਰਾ][ਮਿੰਟ] *[Pandrā][mint]* *[Fifteen][minutes]* Resultant nodes: [ਪੰਦਰਾ] *[Pandrā]* *[Fifteen]*
3.	(MINUTES ,%a)({ STAIL |CTAIL}, ^SECOND S, ^COLON1, ^COLON2, %c):=(%a)([[00]],[00], "00", SECONDS) (%c);	Here, *%a* refers to node [ਪੰਦਰਾ], *%c* refers to STAIL. This rule adds a node [00] after node *%a i.e.* before STAIL as shown in Action Taken column and is given an attribute 'SECONDS'.	Original nodes: [ਪੰਦਰਾ] *[Pandrā]* *[Fifteen]* Resultant nodes: [ਪੰਦਰਾ][00] *[Pandrā][00]* *[Fifteen][00]*
4.	({SHEAD| CHEAD},	Here, *%a* refers to SHEAD, and *%b* refers to node [ਪੰਦਰਾ].	Original nodes: [ਪੰਦਰਾ][00]

	^HOURS, ^COLON1, ^COLON2, %a)(MINUTES, %b):=(%a)([[00]],[00], "00", HOURS) (%b);	This rule adds a node [00] after SHEAD and before *%b* as shown in Action Taken column. This new node is given an attribute 'HOURS'.	[*Pandrā*][00] [*Fifteen*][00] Resultant nodes: [00][ਪੰਦਰਾ][00] *[00][Pandrā][00]* *[00][Fifteen][00]*
5.	(HOURS, %a)(MINUTES, %b)(SECONDS, %c):=(%a) ([[:]],[:],":", COLON1, %x)(%b)([[:]],[:],":", COLON2, %y)(%c);	Here, *%a* refers to node [00] having attribute 'HOURS', *%b* refers to node [ਪੰਦਰਾ], and *%c* refers to node [00] having attribute 'SECONDS'. This rule adds node [:], being referred to as *%x*, between *%a* and *%b*, and node [:], being referred to as *%y*, between *%b* and *%c*. The node *%x* is given an attribute 'COLON1', while node *%y* is given attribute 'COLON2'.	Original nodes: [00][ਪੰਦਰਾ][00] *[00][Pandrā][00]* *[00][Fifteen][00]* Resultant nodes: [00][:][ਪੰਦਰਾ][:][00] *[00][:][Pandrā][:][00]* *[00][:][Fifteen][:][00]*
6.	(HOURS, %a)(COLON1, %b) (MINUTES ,%c) (COLON2, %d) (SECONDS ,%e):=(%a	Here, *%a* refers to node [00] having attribute 'HOURS', *%b* refers to node [:] having attribute'COLON1', *%c* refers to node [ਪੰਦਰਾ], *%d* refers to node [:] having attribute 'COLON2', and *%d* refers to node [00] having attribute 'SECONDS'. This rule	Original nodes: [00][:][ਪੰਦਰਾ][:][00] *[00][:][Pandrā][:][00]* *[00][:][Fifteen][:][00]* Resultant nodes: [[00][:][ਪੰਦਰਾ][:][00]] *[[00][:][Pandrā][:][00]]* *[[00][:][Fifteen][:][00]]*

	&%b&%c &%d&%e, -COLON1, -COLON2, -MINUTES -SECONDS ,-HOURS, -MYTIME, -TIME_DIGIT, -HOURSMZ ,-MINUTESMZ);	concatenates all nodes to form a new hyper node as shown in Action Taken column. All the surplus attributes are removed from this hyper node.	Now the natural language word 'ਪੰਦਰਾ' is replaced by its universal word and final output is generated by IAN as shown in (4.37).

UNL generated by IAN for sentence (4.35) is given in (4.37).

{org}
ਪੰਦਰਾ ਮਿੰਟ

{/org}
{unl}
[w] "00:15:00":08 [/w]
{/unl} ...(4.37)

Here, ':08' is the scope internally generated by the IAN tool.

4.7.8 UNLization of Verbs

Verbs are an important constituent of any natural language. They describe an action or occurrence or indicate a state of being like 'run', 'eat', 'sleeping' *etc*. The UNLization process for verbs has been presented using a simple example sentence (4.38).

ਉਸਨੇ ਉਹਨੂੰ ਕੱਲ੍ਹ ਮਾਰ ਦਿੱਤਾ ...(4.38)
usnē uhnūṃ kallh mār dittā
He killed him yesterday

After the tokenization of example sentence given in (4.38) with IAN tool, seven lexical items are identified as shown in (4.39).

[ਉਸਨੇ]{}"00.@3.@male"(LEX=R,POS=PPR,CAS=NOM,PER=3PS)<pan,0,0>;

[ਉਹਨੂੰ]{}"00.@3.@male" (LEX=R,POS=PPR,CAS=NOM,PER=3PS)<pan,0,0>;

[ਕੱਲ੍ਹ]{}"yesterday"(LEX=A,POS=AAV,SEM=TME)<pan,0,0>;

[ਮਾਰ ਦਿੱਤਾ]{}"kill"(LEX=V,POS=VER,TRA=TSTD,GEN=MCL,ATE=PAS,NUM

=SNG)<pan,0,0>;

Three blank spaces are also identified as :-

[ ]{}""(BLK)<pan,0,0>; ...(4.39)

Here, "*V*" and "*R*" indicate that lexical category of those nodes are verb and pronoun respectively, "*TRA=TSTD"* indicates that verb is direct transitive, "*NPR*" means that pronoun is an indefinite pronoun whereas "*COP*" means that the verb is a linking verb which is used to link the subject of a sentence with a predicate. "*SEM=TME*" means that semantics of a node is time. The process of UNLization of example sentence (4.38) has been illustrated in Table 4.15.

Table 4.15: UNLization Process for Example Sentence (4.38)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK):=;	Here, *%a* refers to blank node []. This rule is fired thrice and deletes all the blank spaces.	Original nodes : [ਉਸਨੇ][][ਉਹਨੂੰ][][ਕੱਲ੍ਹ][] [ਮਾਰ ਦਿੱਤਾ] *[usnē][][uhnūṃ][]* *[kallh][][mār dittā]* Resultant nodes: [ਉਸਨੇ][ਉਹਨੂੰ][ਕੱਲ੍ਹ] [ਮਾਰ ਦਿੱਤਾ] *[usnē][uhnūṃ][kallh]* *[mār dittā]*
2.	(V,PAS,^@past, %a):=(%a,+att= @past);	Here, *%a* refers to node [ਮਾਰ ਦਿੱਤਾ]. This rule gives	Original nodes : [ਉਸਨੇ][ਉਹਨੂੰ][ਕੱਲ੍ਹ]

		an attribute '@past' to node *%a*.	[ਮਾਰ ਦਿੱਤਾ] *[usnē][uhnūm][kallh] [mār dittā]*
			Resultant nodes: [ਉਸਨੇ][ਉਹਨੂੰ][ਕੱਲ੍ਹ] [ਮਾਰ ਦਿੱਤਾ @past] *[usnē][uhnūṃ][kallh] [mār dittā@past]*
3.	(A,%a)(V,TRA =TSTD,%b):=(NA(%b;%a),+V ,+TRA=TSTD,+ TIM);	Here, *%a* refers to node [ਕੱਲ੍ਹ], *%b* refers to node [ਮਾਰ ਦਿੱਤਾ@past]. This rule creates a relation 'NA' between nodes *%b*, and *%a* as its first and second argument respectively.	Original nodes: [ਉਸਨੇ][ਉਹਨੂੰ][ਕੱਲ੍ਹ] [ਮਾਰ ਦਿੱਤਾ@past] *[usnē][uhnūm][kallh] [mār dittā@past]* Resultant nodes: [ਉਸਨੇ][ਉਹਨੂੰ][NA([ਮਾਰ ਦਿੱਤਾ@past];[ਕੱਲ੍ਹ])] *[usnē][uhnūm][NA([mār dittā];[kallh])]*
4.	(R,PPR,%a)(V, TSTD,%b):=(NA(%b;%a),+ OBJ,V,TSTD, +GEN=%b, +NUM=%b);	Here, *%a* refers to node [ਉਹਨੂੰ], *%b* refers to node [NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])] . This rule resolves a relation 'NA' with *%b* and *%a* as first and second argument respectively. The final node is given attribute 'OBJ' and this node is	Original nodes: [ਉਸਨੇ][ਉਹਨੂੰ][NA([ਮਾਰ ਦਿੱਤਾ@past];[ਕੱਲ੍ਹ])] *[usnē][uhnūm][NA([mār dittā@past]; [kallh])]* Resultant nodes: [ਉਸਨੇ] [NA([NA(

		treated as verb.	[ਮਾਰ ਦਿੱਤਾ @past]; [ਕੱਲ੍ਹ])]; [ਉਹਨੂੰ])] *[usnē]* [NA([NA([*mār dittā*@past]; [*kallh*])];[*uhnūṃ*])]
5.	({SHEAD| CHEAD},%z) (R,{CPR|PPR}, %a)(V,TSTD, %b):=(NA(%b; %a),+AGT,V, TSTD);	Here, *%a* refers to SHEAD, *%b* refers to node [ਉਸਨੇ], and *%c* refers to node [NA([NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])]; [ਉਹਨੂੰ])] . This rule resolves a relation 'NA' with *%b* and *%a* as first and second argument respectively. The final node is given attribute 'AGT' and this node is treated as verb.	Original nodes: [ਉਸਨੇ][NA([NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])]; [ਉਹਨੂੰ])] *[usnē]* [NA([NA([*mār dittā* @past]; [*kallh*])]; [*uhnūṃ*])] Resultant nodes: [NA([NA([NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])]; [ਉਹਨੂੰ])]; [ਉਸਨੇ])] *[NA([NA([NA ([mār dittā@past] ;[kallh])];[uhnūṃ])]; [usnē])]*
6.	(NA(NA(%a;%b),OBJ,%w;%c), AGT,%r):=(%w),(NA(%a;%c), %e,+AGT);	Here, *%a* refers to node [NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])], *%b* refers to node [ਉਹਨੂੰ], *%c* refers to node [ਉਸਨੇ], *%w* [NA([NA(	Original nodes: [NA([NA([NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])];[ਉਹਨੂੰ])];[ਉਸਨੇ])] *[NA([NA([NA([mār dittā @past]; [kallh])];[uhnūm])];*

		[ਮਾਰ ਦਿੱਤਾ@past];[ਕੱਲ੍ਹ])]; [ਉਹਨੂੰ])] , and *%r* refers to node [NA([NA([NA([ਮਾਰ ਦਿੱਤਾ@past];[ਕੱਲ੍ਹ])]; [ਉਹਨੂੰ])];[ਉਸਨੇ])]. This rule splits the node into two nodes *%w* and *%e*. Node *%e* holds relation NA with *%a* and *%c* as its first and second argument respectively.	*[usnē])]* Resultant nodes: Node 1: [NA([NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])];[ਉਹਨੂੰ])] *[NA([NA([mār dittā@past]; [kallh])];[uhnūṃ])]* Node 2: [NA([NA([ਮਾਰ ਦਿੱਤਾ@past];[ਕੱਲ੍ਹ])] ;[ਉਸਨੇ])] *[NA([NA([mār dittā@ past]; [kallh])];[usnē])]*
7.	(NA(NA(%a;%b),TIM,%w;%c), OBJ,%r):=(%w) ,(NA(%a;%c), %k,+OBJ);	Here, *%a* refers to node [ਮਾਰ ਦਿੱਤਾ@past], *%b* refers to node [ਕੱਲ੍ਹ], *%w* refers to node [NA([ਮਾਰ ਦਿੱਤਾ@past] ;[ਕੱਲ੍ਹ])], *%c* refers to node [ਉਹਨੂੰ], and *%r* refers to node [NA([NA([ਮਾਰ ਦਿੱਤਾ @past];[ਕੱਲ੍ਹ])];[ਉਹਨੂੰ])]. This rule splits a node into *%w* and *%k*. Node *%k* holds	Original Nodes: [NA([NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])];[ਉਹਨੂੰ])] *[NA([NA([mār dittā@past]; [kallh])];[uhnūm])]* Resultant Nodes: Node 1: [NA([ਮਾਰ ਦਿੱਤਾ@past] ;[ਕੱਲ੍ਹ])]

		relation NA with *%a* and *%c* as its first and second argument respectively.	*[NA([mār dittā@past] ;[kallh])]* Node 2: [NA([ਮਾਰ ਦਿੱਤਾ@past] ;[ਉਹਨੂੰ])] *[NA([mār dittā@past] ;[uhnūm])]*
8.	(NA(NA(%a;%b),TIM,%w;%c), AGT,%r):=(%w),(NA(%a;%c), %k,+AGT);	Here, *%a* refers to node [ਮਾਰ ਦਿੱਤਾ@past], *%b* refers to node [ਕੱਲ੍ਹ], *%w* refers to node [NA([ਮਾਰ ਦਿੱਤਾ@past] ;[ਕੱਲ੍ਹ])], *%c* refers to node [ਉਸਨੇ], and *%r* refers to node [NA([NA([ਮਾਰ ਦਿੱਤਾ @past];[ਕੱਲ੍ਹ])];[ਉਸਨੇ])]. This rule splits a node into *%w* and *%k.* Node *%k* holds relation NA with *%a* and *%c* as its first and second argument respectively.	Original Nodes: [NA([NA([ਮਾਰ ਦਿੱਤਾ@past]; [ਕੱਲ੍ਹ])];[ਉਸਨੇ])] *[NA([NA(* *[mār dittā @past];* *[kallh])];[usnē])* Resultant Nodes: Node 1: [NA([ਮਾਰ ਦਿੱਤਾ@past] ;[ਕੱਲ੍ਹ])] *[NA([mār dittā@past] ;[kallh])]* Node 2: [NA([ਮਾਰ ਦਿੱਤਾ @past];

			[ਉਸਨੇ])] *[NA([mār dittā@past];[usnē])]*
9.	(NA(%a;%b), AGT):=agt(%a; %b);	Here, *%a* refers to node [NA(ਮਾਰ ਦਿੱਤਾ@past)], and *%b* refers to node [ਉਸਨੇ]. This rule changes the name of relation from 'NA' to 'agt' as required in the final UNL.	Original Nodes: [NA([ਮਾਰ ਦਿੱਤਾ@past];[ਉਸਨੇ])] *[NA([mār dittā@past];[usnē])]* Resultant Nodes: [agt([ਮਾਰ ਦਿੱਤਾ@past];[ਉਸਨੇ])] *[agt([mār dittā@past];[usnē])]*
10.	(NA(%a;%b), OBJ):=obj(%a; %b);	Here, *%a* refers to node [NA(ਮਾਰ ਦਿੱਤਾ@past)], and *%b* refers to node [ਉਹਨੂੰ]. This rule changes the name of relation from 'NA' to 'obj' as required in the final UNL.	Original Nodes: [NA([ਮਾਰ ਦਿੱਤਾ@past];[ਉਹਨੂੰ])] *[NA([mār dittā@past];[uhnūṃ])]* Resultant Nodes: [obj([ਮਾਰ ਦਿੱਤਾ@past];[ਉਹਨੂੰ])] *[obj([mār dittā@past];[uhnūm])]*
11.	(NA(%a;%b), TIM):=tim(%a; %b);	Here, *%a* refers to node [NA(ਮਾਰ ਦਿੱਤਾ@past)], and	Original Nodes: [NA([ਮਾਰ ਦਿੱਤਾ@past]

		%b refers to node [ਕੱਲ੍ਹ]. This rule changes the name of relation from 'NA' to 'tim' as required in the final UNL.	;[ਕੱਲ੍ਹ])] *[NA([mār dittā@past]; [kallh])]* Resultant Nodes: [tim([ਮਾਰ ਦਿੱਤਾ@past] ;[ਕੱਲ੍ਹ])] *[tim([mār dittā@past]; [kallh])]* Now all the natural language words are replaced by their universal words and final output is generated by IAN as shown in (4.40).

UNL generated by IAN for sentence (4.38) is specified in (4.40) and its UNL graph is shown in Figure 4.11.

{org}

ਉਸਨੇ ਉਹਨੂੰ ਕੱਲ੍ਹ ਮਾਰ ਦਿੱਤਾ

{/org}

{unl}

agt(kill.@past, 00:01.@3.@male)

obj(kill.@past, 00:03.@3.@male)

tim(kill.@past, yesterday)

{/unl} ...(4.40)

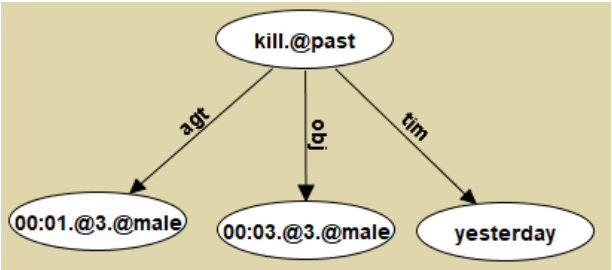


Figure 4.11: UNL Graph of (4.40)

4.7.9 UNLization of Nouns and Adjectives

The main role of an adjective is to assign attributes to a noun. Adjectives are different from determiners, which express references rather than qualities. The UNLization process for nouns and adjectives has been presented using a simple example sentence (4.41).

ਇਕ ਸੋਹਣੀ ਗੱਡੀ, ਇਕ ਮਹਿੰਗੀ ਗੱਡੀ ਅਤੇ ਇਕ ਨਵਾਂ ਪਿਆਲਾ ...(4.41)

ik sōhṇī gaḍḍī, ik mahiṅgī gaḍḍī atē ik navāṃ piālā

A beautiful car, a expensive car and a new mug.

After the tokenization of example sentence given in (4.41) with IAN tool, twenty lexical items are identified as shown in (4.42).

[ਸੋਹਣੀ]{}"beautiful"(LEX=J,POS=ADJ,GEN=FEM)<pan,0,0>;

[ਗੱਡੀ]{}"car"(LEX=N,POS=NOU,NUM=SNG)<p an,0,0>;

[,]{}""(LEX=C,POS=COO,rel=and)<pan,0,0>;

[ਮਹਿੰਗੀ]{}"expensive"(LEX=J,POS=ADJ)<pan,0,0>;

[ਗੱਡੀ]{}"car"(LEX=N,POS=NOU,NUM=SNG) <pan,0,0>;

[ਅਤੇ]{}"and"(LEX=C,POS=COO,rel=and)<pan,0,0>;

[ਨਵਾਂ]{}"new"(LEX=J,POS=ADJ)<pan,0,0>;

[ਪਿਆਲਾ]{}"mug"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>;

Three nodes are identified as :-

[ਇਕ]{}""(LEX=D,POS=ART,att=@indef)<pan,0,0>;

Nine blank spaces are also identified as :-

[]{}""(BLK)<pan,0,0>; ...(4.42)

Here, "*J*" and "*ADJ*" represent lexical category and part of speech respectively as adjective, "*FEM*" represents gender of the node as female, and "*ART*" indicates that determiner is an article. Articles are used to express definiteness like 'a', 'the' *etc*. The process of UNLization of example sentence (4.41) has been illustrated in Table 4.16.

Table 4.16: UNLization Process for Example Sentence (4.41)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK): =;	Here, *%a* refers to blank node []. This rule is fired nine times and deletes all the blank spaces.	Original nodes : [ਇਕ][][ਸੋਹਣੀ][][ਗੱਡੀ][,][] [ਇਕ][][ਮਹਿੰਗੀ][][ਗੱਡੀ][] [ਅਤੇ][][ਇਕ][][ਨਵਾਂ][] [ਪਿਆਲਾ] *[ik][][sōhnī][][gaddī][,][]* *[ik][][mahiṅgī][][gaddī][]* *[atē][][ik][][navām][]* *[piālā]* *[A][][beautiful][][car][,]* *[][a][expensive][][car][]* *[and][][a][][new][][mug]* Resultant nodes : [ਇਕ][ਸੋਹਣੀ][ਗੱਡੀ][,][ਇਕ] [ਮਹਿੰਗੀ][ਗੱਡੀ][ਅਤੇ][ਇਕ] [ਨਵਾਂ][ਪਿਆਲਾ] *[ik][sōhnī][gaddī][,][ik]* *[mahiṅgī][gaddī][atē][ik]* *[navām][piālā]*

			[A][beautiful][car][,][a] [expensive][car][and][a] [new][mug]
2.	(D,att,%a)(J ,%b)(N,%c) :=(NA(%c, +att=%a; %b),+N, +NOU, +MOD);	Here, *%a* refers to node [ਇਕ], *%b* refers to node [ਸੋਹਣੀ], and node *%c* refers to [ਗੱਡੀ]. This rule resolves a relation 'NA' whose first and second argument are *%c* and *%b* respectively. The attributes of node *%a* are given to the first argument of 'NA' relation. This new node is given attributes 'N', 'NOU', 'MOD'.	Original nodes : [ਇਕ][ਸੋਹਣੀ][ਗੱਡੀ][,][ਇਕ] [ਮਹਿੰਗੀ][ਗੱਡੀ][ਅਤੇ][ਇਕ] [ਨਵਾਂ][ਪਿਆਲਾ] *[ik][sōhṇī][gaḍḍī][,][ik] [mahiṅgī][gaḍḍī][atē][ik] [navāṃ][piālā]* *[A][beautiful][car][,][a] [expensive][car][and][a] [new][mug]* Resultant nodes: [NA([ਗੱਡੀ@indef]; [ਸੋਹਣੀ])][,][ਇਕ][ਮਹਿੰਗੀ] [ਗੱਡੀ][ਅਤੇ][ਇਕ][ਨਵਾਂ] [ਪਿਆਲਾ] *[NA([gaḍḍī@indef]; [sōhṇī])][,][ik][mahiṅgī] [gaḍḍī][atē][ik][navāṃ] [piālā]* *[NA([car@indef]; [beautiful])][,][a] [expensive][car][and][a] [new][mug]*

3.	(D,att,%a)(J ,%b)(N,%c) :=(NA(%c, +att=%a; %b),+N, +NOU, +MOD);	Here, *%a* refers to node [ਇਕ], *%b* refers to node [ਮਹਿੰਗੀ], and node *%c* refers to [ਗੱਡੀ]. This rule resolves a relation 'NA' whose first and second argument are *%c* and *%b* respectively. The attributes of node *%a* are given to the first argument of 'NA' relation. This new node is given attributes 'N', 'NOU', 'MOD'.	Original nodes: [NA([ਗੱਡੀ@indef]; [ਸੋਹਣੀ])][,][ਇਕ][ਮਹਿੰਗੀ] [ਗੱਡੀ][ਅਤੇ][ਇਕ][ਨਵਾਂ] [ਪਿਆਲਾ] *[NA([gaḍḍī@indef];* *[sōhnī])][,][ik][mahiṅgī]* *[gaddī][atē][ik][navām]* *[piālā]* *[NA([car@indef];* *[beautiful])][,][a]* *[expensive][car][and][a]* *[new][mug]* Resultant nodes: [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])][,][NA([ਗੱਡੀ@indef]; [ਮਹਿੰਗੀ])][ਅਤੇ][ਇਕ][ਨਵਾਂ] [ਪਿਆਲਾ] *[NA([gaddī@indef];* *[sōhnī])][,][NA(* *[gaḍḍī@indef];[mahiṅgī])]* *[atē][ik][navām][piālā]* *[NA([car@indef];* *[beautiful])][,][NA(* *[car@indef];[expensive]))]* *[and][a][new][mug]*

4.	(N,NOU, %a)(C,%b)(N,NOU,%c):=(NA(%c; %a),+N, +NOU, +AND);	Here, *%a* refers to node [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])], *%b* refers to node [,] and *%c* refers to node [NA([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]. This rule resolves a relation 'NA' whose first and second argument are *%c* and *%a* respectively. This new node so formed is given an attribute 'N', 'NOU', and 'AND'.	Original nodes: [NA([ਗੱਡੀ@indef]; [ਸੋਹਣੀ])][,][NA([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])] [ਅਤੇ][ਇਕ][ਨਵਾਂ][ਪਿਆਲਾ] *[NA([gaḍḍī@indef]; [sōhnī])][,][NA([gaddī@indef];[mahiṅgī])] [atē][ik][navām][piālā]* *[NA([car@indef]; [beautiful])][,][NA([car@indef];[expensive])] [and][a][new][mug]* Resultant nodes: [NA([NA([ਗੱਡੀ@indef]; [ਮਹਿੰਗੀ])];[NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])] [ਅਤੇ][ਇਕ][ਨਵਾਂ][ਪਿਆਲਾ] *[NA([NA([gaddī@indef]; [mahiṅgī])];[NA([gaḍḍī@indef];[sōhnī])])] [atē][ik][navām][piālā]* *[NA([NA([car@indef]; [expensive])];[NA([car@indef];[beautiful])])] [and][a][new][mug]*

5.	(D,att,%a)(J ,%b)(N,%c) :=(NA(%c, +att=%a; %b),+N, +NOU, +MOD);	Here, *%a* refers to node [ਇਕ], *%b* refers to node [ਨਵਾਂ], and node *%c* refers to [ਪਿਆਲਾ]. This rule resolves a relation 'NA' whose first and second argument are *%c* and *%b* respectively. The attributes of node *%a* are given to the first argument of 'NA' relation. This new node is given attributes 'N', 'NOU', 'MOD'.	Original nodes: [NA([NA([ਗੱਡੀ@indef]; [ਮਹਿੰਗੀ])];[NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])] [ਅਤੇ][ਇਕ][ਨਵਾਂ][ਪਿਆਲਾ] *[NA([NA([gaḍḍī@indef]; [mahiṅgī])];[NA([gaḍḍī @indef];[sōhnī])])][atē] [ik][na vām][piālā]* *[NA([NA([car@indef]; [expensive])];[NA([car@indef];[beautiful])])] [and][a][new][mug]* Resultant nodes: [NA([NA([ਗੱਡੀ@indef]; [ਮਹਿੰਗੀ])];[NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])] [ਅਤੇ][NA([ਪਿਆਲਾ@indef]; [ਨਵਾਂ])] *[NA([NA([gaḍḍī@indef]; [mahiṅgī])];[NA([gaddī@indef];[sōhnī])])] [atē][NA([piālā@indef]; [navām])]* *[NA([NA([car@indef]; [expensive])];[NA(*

			[car@indef];[beautiful])])] [and][NA([mug@indef]; [new])]
6.	(N,NOU, %a)(C,%b)(N,NOU,%c):=(NA(%c; %a),+N, +NOU, +AND);	Here, *%a* refers to node [NA([NA([ਗੱਡੀ@indef]; [ਮਹਿੰਗੀ])];[NA([ਗੱਡੀ@indef]; [ਸੋਹਣੀ])])], *%b* refers to node [ਅਤੇ] and *%c* refers to node [NA([ਪਿਆਲਾ@indef];[ਨਵਾਂ])]. This rule resolves a relation 'NA' whose first and second argument are *%c* and *%a* respectively. This new node so formed is given an attribute 'N', 'NOU', and 'AND'.	Original nodes: [NA([NA([ਗੱਡੀ@indef]; [ਮਹਿੰਗੀ])];[NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])] [ਅਤੇ][NA([ਪਿਆਲਾ@indef]; [ਨਵਾਂ])] *[NA([NA([gaḍḍī@indef]; [mahiṅgī])];[NA([gaḍḍī@indef];[sōhnī])])] [atē][NA([piālā@indef]; [navāṃ])]* *[NA([NA([car@indef]; [expensive])];[NA([car@indef];[beautiful])])] [and][NA([mug@indef]; [new])]* Resultant nodes: [NA([NA([ਪਿਆਲਾ@indef]; [ਨਵਾਂ])];[NA([NA([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])]

			[NA([NA([piālā@indef]; [navām])];[NA([NA([gaḍḍī@indef];[mahiṅgī])] ;[NA([gaḍḍī@indef]; [sōhṇī])])])] *[NA([NA([mug@indef]; [new])];[NA([NA([car@indef];[expensive])]; [NA([car@indef]; [beautiful])])])]*
7.	(NA(%a; %b),MOD): =mod(%a; %b);	Here, *%a* refers to node [ਪਿਆਲਾ@indef], *%b* refers to [ਨਵਾਂ]. This rule changes the name of relation from 'NA' to 'mod' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: [NA([NA([ਪਿਆਲਾ@indef]; [ਨਵਾਂ])];[NA([NA([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[NA([NA([piālā@indef]; [navām])];[NA([NA([gaḍḍī@indef];[mahiṅgī])] ;[NA([gaḍḍī@indef]; [sōhṇī])])])]* *[NA([NA([mug@indef]; [new])];[NA([NA([car@indef];[expensive])]; [NA([car@indef]; [beautiful])])])]* Resultant nodes: [NA([mod([ਪਿਆਲਾ@indef]

			;[ਨਵਾਂ])];[NA([NA([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[NA([mod([piālā@indef]; [navām])];[NA([NA([gaḍḍī@indef];[mahiṅgī])] ;[NA([gaddī@indef]; [sōhnī])])])]* *[NA([mod([mug@indef]; [new])];[NA([NA([car@indef];[expensive])]; [NA([car@indef]; [beautiful])])])]*
8.	(NA(%a; %b),MOD): =mod(%a; %b);	Here, *%a* refers to node [ਗੱਡੀ@indef], *%b* refers to [ਸੋਹਣੀ]. This rule changes the name of relation from 'NA' to 'mod' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: [NA([mod([ਪਿਆਲਾ@indef] ;[ਨਵਾਂ])];[NA([NA([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[NA([mod([piālā@indef]; [navām])];[NA([NA([gaḍḍī@indef];[mahiṅgī])] ;[NA([gaddī@indef]; [sōhnī])])])]* *[NA([mod([mug@indef]; [new])];[NA([NA(*

			[car@indef];[expensive])]; *[NA([car@indef];* *[beautiful])])])]* Resultant nodes: [NA([mod([ਪਿਆਲਾ@indef] ;[ਨਵਾਂ])];[NA([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[NA([mod([piālā@indef];* *[navām])];[NA([mod(* *[gaddī@indef];[mahiṅgī])]* *;[NA([gaḍḍī@indef];* *[sōhṇī])])])]* *[NA([mod([mug@indef];* *[new])];[NA([mod(* *[car@indef];[expensive])];* *[NA([car@indef];* *[beautiful])])])]*
9.	(NA(%a; %b),MOD): =mod(%a; %b);	Here, *%a* refers to node [ਗੱਡੀ@indef], *%b* refers to [ਮਹਿੰਗੀ]. This rule changes the name of relation from 'NA' to 'mod' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: [NA([mod([ਪਿਆਲਾ@indef] ;[ਨਵਾਂ])];[NA([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [NA([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[NA([mod([piālā@indef];*

			[navām])];[NA([mod([gaḍḍī@indef];[mahiṅgī])] ;[NA([gaḍḍī@indef]; [sōhnī])])])] *[NA([mod([mug@indef]; [new])];[NA([mod([car@indef];[expensive])]; [NA([car@indef]; [beautiful])])])]* Resultant nodes: [NA([mod([ਪਿਆਲਾ@indef] ;[ਨਵਾਂ])];[NA([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [mod([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[NA([mod([piālā@indef]; [navām])];[NA([mod([gaḍḍī@indef];[mahiṅgī])] ;[mod([gaḍḍī@indef]; [sōhṇī])])])]* *[NA([mod([mug@indef]; [new])];[NA([mod([car@indef];[expensive])]; [mod([car@indef]; [beautiful])])])]*

10.	(NA(%a; %b),AND): =and(%a; %b);	Here, *%a* refers to node [mod([ਪਿਆਲਾ@indef];[ਨਵਾਂ])], and *%b* refers to node [NA([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [mod([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])]. This rule changes the name of relation from 'NA' to 'and' keeping same arguments as in original node, as required in the final UNL.	Original nodes: [NA([mod([ਪਿਆਲਾ@indef];[ਨਵਾਂ])];[NA([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [mod([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[NA([mod([piālā@indef];[navām])];[NA([mod([gaḍḍī@indef];[mahiṅgī])];[mod([gaḍḍī@indef];[sōhnī])])])]* *[NA([mod([mug@indef];[new])];[NA([mod([car@indef];[expensive])];[mod([car@indef];[beautiful])])])]* Resultant nodes: [and([mod([ਪਿਆਲਾ@indef];[ਨਵਾਂ])];[NA([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [mod([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[and([mod([piālā@indef];[navām])];[NA([mod([gaḍḍī@indef];[mahiṅgī])]*

			;[mod([gaḍḍī@indef]; [sōhṇī])])])] *[and([mod([mug@indef]; [new])];[NA([mod([car@indef];[expensive])]; [mod([car@indef]; [beautiful])])])]*
11.	(NA(%a; %b),AND): =and(%a; %b);	Here, *%a* refers to node[mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])], and *%b* refers to node [mod([ਗੱਡੀ@indef];[ਸੋਹਣੀ])]. This rule changes the name of relation from 'NA' to 'and' keeping same arguments as in the original node, as required in the final UNL.	Original nodes: [and([mod([ਪਿਆਲਾ@indef] ;[ਨਵਾਂ])];[NA([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [mod([ਗੱਡੀ@indef];[ਸੋਹਣੀ])])])] *[and([mod([piālā@indef]; [navām])];[NA([mod([gaḍḍī@indef];[mahiṅgī])] ;[mod([gaḍḍī@indef]; [sōhṇī])])])]* *[and([mod([mug@indef]; [new])];[NA([mod([car@indef];[expensive])]; [mod([car@indef]; [beautiful])])])]* Resultant nodes: [and([mod([ਪਿਆਲਾ@indef] ;[ਨਵਾਂ])];[and([mod([ਗੱਡੀ@indef];[ਮਹਿੰਗੀ])]; [mod([ਗੱਡੀ@indef];[ਸੋਹਣੀ]

			)])])] *[and([mod([piālā@indef]; [navām])];[and([mod([gaḍḍī@indef];[mahiṅgī])] ;[mod([gaḍḍī@indef]; [sōhnī])])])]* *[and([mod([mug@indef]; [new])];[and([mod([car@indef];[expensive])]; [mod([car@indef]; [beautiful])])])]* Now all the natural language words are replaced by their universal words, internal hypernodes are represented by their scopes as shown in final output generated by IAN as given in (4.43).

UNL generated by IAN for example sentence (4.41) is specified in (4.43) and its UNL graph is shown in Figure 4.12.

{org}
ਇਕ ਸੋਹਣੀ ਗੱਡੀ, ਇਕ ਮਹਿੰਗੀ ਗੱਡੀ ਅਤੇ ਇਕ ਨਵਾਂ ਪਿਆਲਾ

{/org}
{unl}
and(:06, :09)
mod:06(mug:0L.@indef, new:0J)

and:09(:07, :08)

mod:07(car:0D.@indef, expensive:0B)

mod:08(car:05.@indef, beautiful:03)

{/unl} ...(4.43)

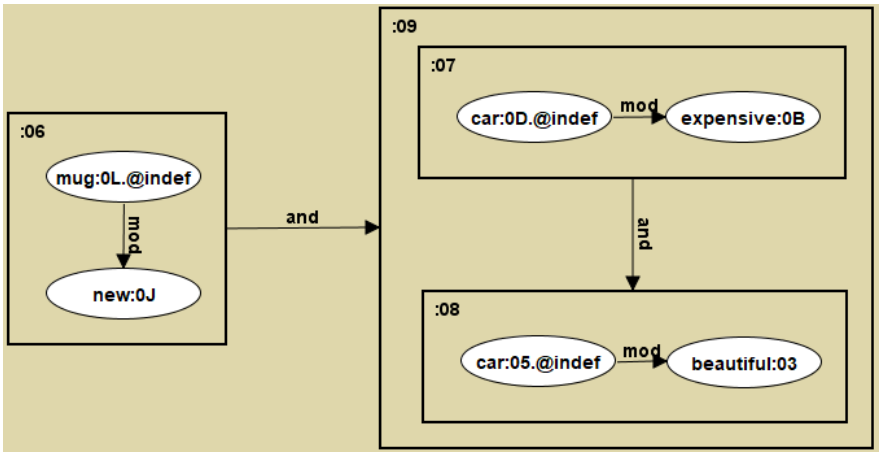


Figure 4.12: UNL Graph of (4.43)

Here, ‘:03’, ‘:05’, ‘:07’, ‘:08’, ‘:09’, ‘:0B’, ‘:0D’, ‘:0L’, ‘:0J’ are the scopes internally generated by the IAN tool.

4.7.10 UNLization of Pronouns

A pronoun is a word or form that substitutes for a noun. For example in the sentence “I did it myself”, ‘myself’ is a pronoun. The UNLization process for pronouns has been presented using a simple example sentence (4.44).

ਉਹ ਉਸਨੂੰ ਪਿਆਰ ਕਰਦਾ ਹੈ ...(4.44)

ōhh usnūṃ piār karadā hai

He loves her

After the tokenization of example sentence given in (4.44) with IAN tool, Seven lexical items are identified as shown in (4.45).

[ਉਹ]{}"00.@3.@male" (LEX=R,POS=PPR,CAS=NOM,PER=3PS)<pan,0,0>;

[ਉਸਨੂੰ]{}"00.@3.@female" (LEX=R,POS=PPR,CAS=OBL,PER=3PS)pan,0,0>;

[ਪਿਆਰ]{}"love"(LEX=V,POS=VER,TRA=TSTD)<pan,0,0>;

[ਕਰਦਾ ਹੈ]{}""(LEX=V,POS=AUX,ATE=PRS,PER=3PS,att=@present,att1=@male, GEN=MCL)<pan,0,0>;

Three blank spaces are also identified as :-

[]{}""(BLK)<pan,0,0>; ...(4.45)

Here, "*LEX=R*" represents lexical category pronoun, "*POS=PPR*" represents part-of-speech as personal pronoun, "*CAS=OBL*" means that case is oblique, and "*PER=3PS*" represents 3rd person. Rest every other tagset has been described in previous examples. The process of UNLization of example sentence (4.44) has been illustrated in Table 4.17.

Table 4.17: UNLization Process for Example Sentence (4.44)

S.No	TRule fired	Description	Action Taken
1.	(%a,BLK): =;	Here, *%a* refers to blank node []. This rule is fired three times and deletes all the blank spaces.	Original nodes : [ਉਹ][][ਉਸਨੂੰ][][ਪਿਆਰ][] [ਕਰਦਾ ਹੈ] *[ōhh][][usnūm][][piār][] [karadā hai]* Resultant nodes : [ਉਹ][ਉਸਨੂੰ][ਪਿਆਰ] [ਕਰਦਾ ਹੈ] *[ōhh][usnūm][piār] [karadā hai]*
2.	(V,VER, TSTD,%a)(V,AUX,%b):=(%a,+att =%b,+GEN =%b, +NUM= %b);	Here, *%a* refers to node [ਪਿਆਰ], and *%b* refers to node [ਕਰਦਾ ਹੈ], This rule deletes node *%b* and gives its attributes to node *%a*.	Original nodes : [ਉਹ][ਉਸਨੂੰ][ਪਿਆਰ] [ਕਰਦਾ ਹੈ] *[ōhh][usnūm][piār] [karadā hai]* Resultant nodes: [ਉਹ][ਉਸਨੂੰ] [ਪਿਆਰ@present]

			[ōhh][usnūm] [piār@present]
3.	(R,PPR,%a) (V,TSTD,%b):=(NA(%b;%a), +OBJ,V, TSTD, +GEN=%b, +NUM= %b);	Here, *%a* refers to node [ਉਸਨੂੰ], and *%b* refers to node [ਪਿਆਰ@present]. This rule resolves a relation 'NA' whose first and second argument are *%b* and *%a* respectively. This new node is given attributes 'V', 'OBJ'.	Original nodes: [ਉਹ][ਉਸਨੂੰ] [ਪਿਆਰ@present] *[ōhh][usnūm] [piār@present]* Resultant nodes: [ਉਹ][NA([ਪਿਆਰ@present] ;[ਉਸਨੂੰ])] *[ōhh][NA([piār@present]; [usnūm])]*
4.	({SHEAD |CHEAD}, %z)(R,{ CPR|PPR}, %a)(V, TSTD,MCL ,%b):=(NA(%b;%a,+att =@male),+ AGT,V, TSTD);	Here, *%z* refers to SHEAD, *%a* refers to node [ਉਹ], and *%c* refers to node [NA([ਪਿਆਰ@present];[ਉਸਨੂੰ])] . This rule resolves a relation 'NA' whose first and second arguments are *%b* and *%a* respectively. This new node is given attributes 'V', 'AGT'.	Original nodes: [ਉਹ][NA([ਪਿਆਰ@present] ;[ਉਸਨੂੰ])] *[ōhh][NA([piār@present]; [usnūm])]* Resultant nodes: [NA([NA([ਪਿਆਰ@present];[ਉਸਨੂੰ])]; [ਉਹ])] *[NA([NA([piār@present]; [usnūm])];[ōhh])]*
5.	(NA(NA(	Here, *%a* refers to node	Original nodes:

	%a;%b), OBJ,%w; %c),AGT, %r):=(%w), (NA(%a;%c),%d,+AGT);	[ਪਿਆਰ@present], *%b* refers to node [ਉਸਨੂੰ], *%w* refers to node [NA([ਪਿਆਰ@present];[ਉਸਨੂੰ])], *%c* refers to [ਉਹ], and *%r* refers to node [NA([NA([ਪਿਆਰ@present]; [ਉਸਨੂੰ])];[ਉਹ])]. This rule splits the node into two nodes *%w* and *%d*. The node *%d* holds a relation 'NA' whose first and second arguments are *%a* and *%c* respectively.	[NA([NA([ਪਿਆਰ@present];[ਉਸਨੂੰ])]; [ਉਹ])] *[NA([NA([piār@present]; [usnūm])];[ōhh])]* Resultant nodes: Node 1: [NA([ਪਿਆਰ@present]; [ਉਸਨੂੰ])] *[NA([piār@present]; [usnūm])]* Node 2: [NA([ਪਿਆਰ@present]; [ਓੱਹ])] *[NA([piār@present]; [ōhh])]*
6.	(NA(%a; %b),AGT): =agt(%a; %b);	Here, *%a* refers to node [ਪਿਆਰ@present], and *%b* refers to node [ਉਹ]. This rule chages the name of relation from 'NA' to 'agt' as required in the final output.	Original nodes: [NA([ਪਿਆਰ@present]; [ਓੱਹ])] *[NA([piār@present]; [ōhh])]* Resultant nodes:

			[agt([ਪਿਆਰ@present]; [ਓੱਹ])] *[agt([piār@present]; [ōhh])]*
7.	(NA(%a; %b),OBJ):= obj(%a;%b) ;	Here, *%a* refers to node [ਪਿਆਰ@present], and *%b* refers to node [ਉਹ]. This rule chages the name of relation from 'NA' to 'obj' as required in the final output.	Original nodes: [NA([ਪਿਆਰ@present]; [ਉਸਨੂੰ])] *[NA([piār@present]; [usnūm])]* Resultant nodes: [obj([ਪਿਆਰ@present]; [ਉਸਨੂੰ])] *[obj([piār@present]; [usnūm])]* Now all the natural language words are replaced by their universal words and final output is generated by IAN as given in (4.46).

UNL generated by IAN for sentence (4.44) is given in (4.46).

{org}

ਉਹ ਉਸਨੂੰ ਪਿਆਰ ਕਰਦਾ ਹੈ

{/org}

{unl}

agt(love:05.@present, 00:01.@3.@male)

obj(love:05.@present, 00:03.@3.@female)

{/unl} ...(4.46)

4.8 Role of X-Bar in UNLization

Ever since the UNL programme was launched, UNLization and NLization had been done by various computational linguists across the globe. It was realized that as the natural language sentences become more and more complex, the number of TRules increases significantly and conflicts arises with the previously made TRules. Thus, there was a need to follow a more systematic approach for UNLization. So, the X-Bar approach is followed by computational linguists working under UNL programme.

The X-Bar theory assumes that certain structural similarities are shared by all human languages comprising of the same basic syntactic structure, recognized as the "*X-Bar*" [172]. The abstract configuration of X-Bar is shown in Figure 4.13 [172].

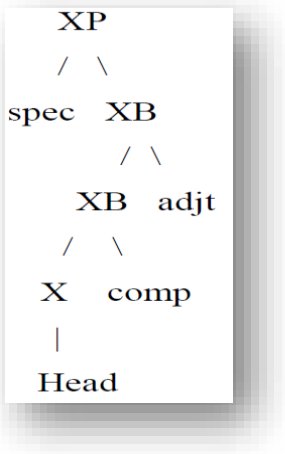


Figure 4.13: X-Bar Abstract Configuration [172]

Here,

- **X** specifies the **head**, *i.e.*, the source or nucleus of the whole syntactic structure, which is actually derived (or projected) out of it. The letter "*X"* indicates an arbitrary lexical category (part of speech). The specific categories are assigned during analysis of a specific utterance. Hence, "*X"* may become "*N"* for a noun, a "*V"* for verb, a "*J*" for adjective, or a "*P*" for preposition.
- **comp** specifies the **complement**. It represents an internal argument and can be a word, clause or phrase. It is important and completes the meaning of head. For example, objects of transitive verbs.
- **adjt** specifies the **adjunct**. It represents a word, clause or phrase which alters the head. It is not as much important and the sentence will be well-formed

grammatically even after removing it. Therefore, adjuncts are expected to be extranuclear.

- **spec** represents **specifier.** It represents an external argument and it can be a word, clause or phrase which determines the head.
- **XB** represents **X-bar**. It is the general name for any of the intermediate projections derived from "*X*".
- **XP** (X-bar-bar, X-double-bar, X-phrase) is the maximal projection of "*X"*.

Consider an example sentence given in (4.47).

The beautiful tortoise won the race. ...(4.47)

X-Bar configuration of example sentence given in (4.47) is shown in Figure 4.14.

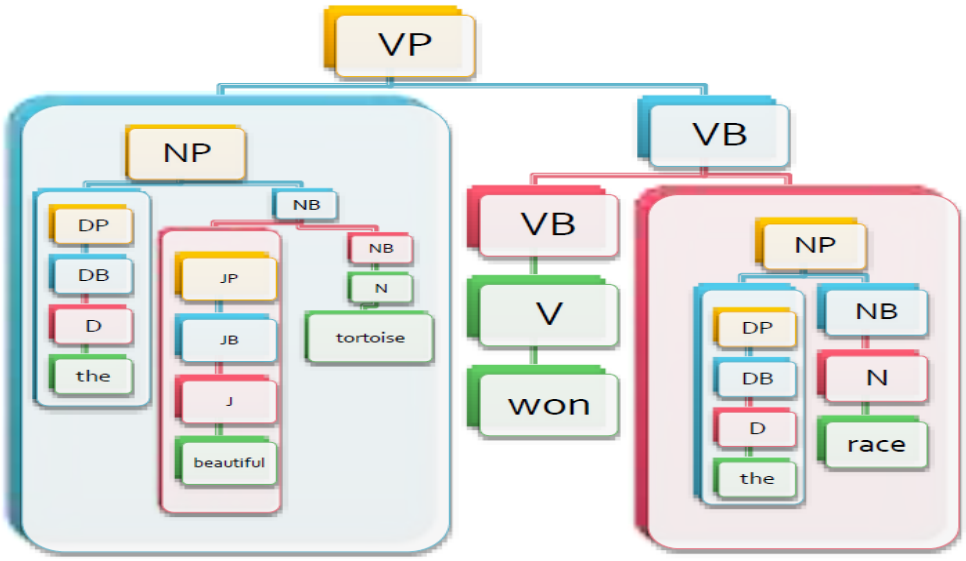


Figure 4.14: X-Bar Structure of Example Sentence (4.47).

In example sentence (4.47), the first determiner "*the*" is promoted up to its maximal projection "*DP*", adjective "*beautiful*" is promoted to its maximal projection "*JP*", and the noun "*tortoise*" is promoted to its intermediate projection "*NB*". The "*JP*" and "*NB*" combine to form the intermediate noun projection "*NB*" which later combines with "*DP*" to form a noun phrase "*NP*".

In this example, consider the substring "*won the race*". Here, verb "*won*" is promoted up to its intermediate projection "*VB*", determiner "*the*" is promoted to its maximal projection "*DP*", and the noun "*race*" is promoted to its intermediate projection "*NB*". The "*DP*" and "*NB*" combine to form the maximal noun projection "*NP*" which later combines with "*VB*" to form a verbal phrase "*VB*", which is in its intermediate

projection. The "*VB*" combines with the maximal projection "*NP*" ("*NP*" is the maximal projection of substring "*the beautiful tortoise*") to form the verbal phrase "*VP*" which is in its maximal projection.

4.9 Transformation (UNLization) using X-Bar Approach

While using X-Bar approach, UNLization is performed in five steps, *i.e.*, parsing, transformation, dearborization, interpretation, and rectification. These are shown in Figure 4.15.

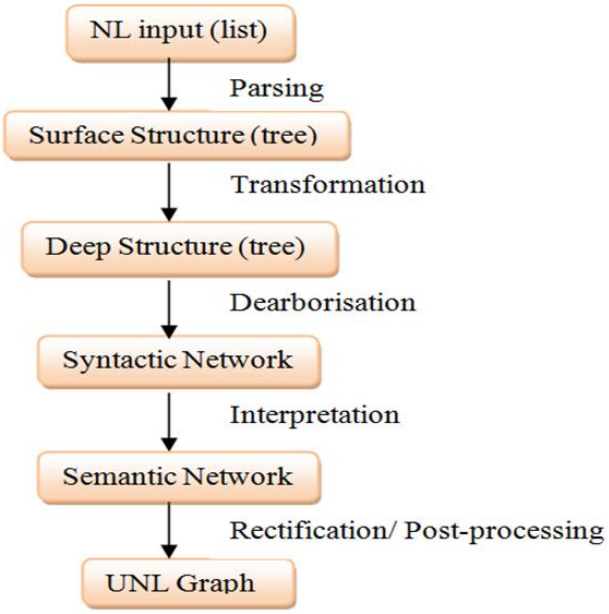


Figure 4.15: UNLization Steps

Each of these five processes are explained as follows.

A. Parsing

When an input document is tokenized by IAN then it is in list form. In parsing, the initial list structure is transformed to tree structure as depicted in Figure 4.16. In parsing, the syntactic analysis of the normalized input is performed.

Figure 4.16: Conversion of List Structure to Tree Structure by Parsing

Consider an example sentence given in (4.48).

John did not kill Mary. ...(4.48)

After tokenization and removing blank spaces the list structure is like [John][did][not][kill][Mary]. After parsing this list structure gets converted to tree structure as shown in Figure 4.17.

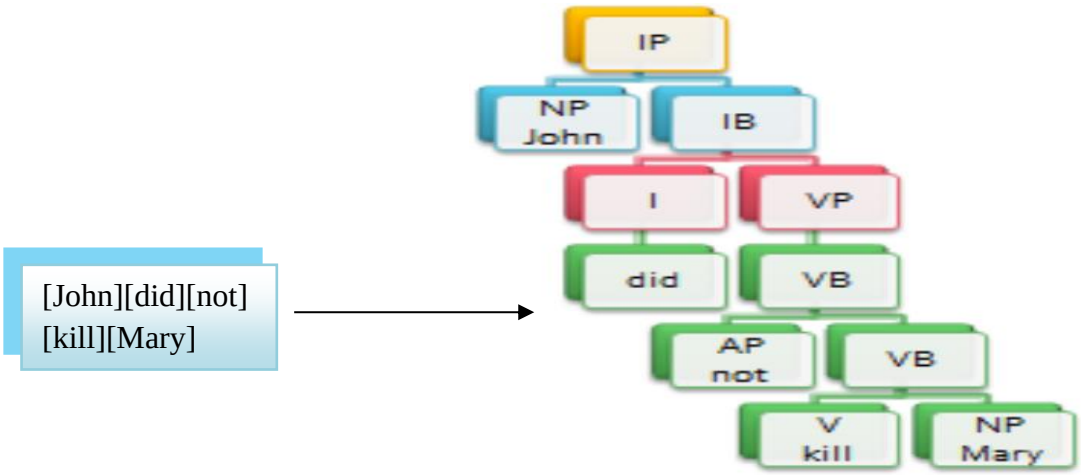


Figure 4.17: Parsing

B. Transformation

The tree which is obtained after parsing is in its surface structure. Some dependency relations that are important in the UNLization process are not presented in the surface structure. For example, consider the sentence "*John did not kill Mary*", the "*NP"* "*John*" is signified at the position of specifier of the "*IP"* "*did not kill Mary*", but it is important to move this "*NP*" to the position of specifier of the "*VP*" "*kill Mary*". In order to do that, the surface structure is converted into a deep structure. The deep syntactic structure is more appropriate to the semantic interpretation. In the transformation phase, this surface tree structure is converted into a modified tree so as to represent its deep syntactic structure as shown in Figure 4.18.

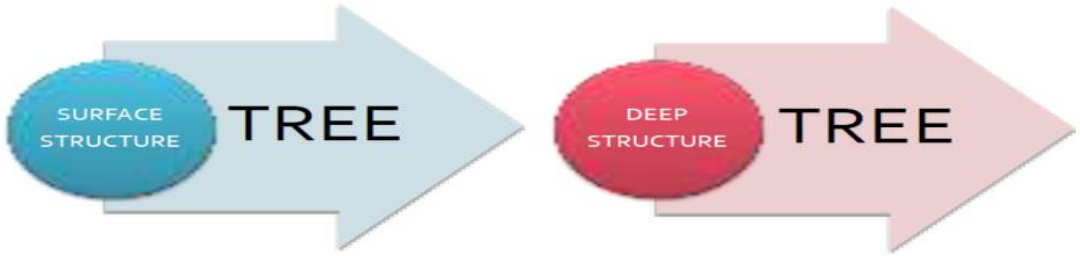


Figure 4.18: Conversion of Surface Structure to Deep Structure

Consider the same example sentence given in (4.48). The surface structure which is obtained after parsing is converted to the deep structure as shown in Figure 4.19.

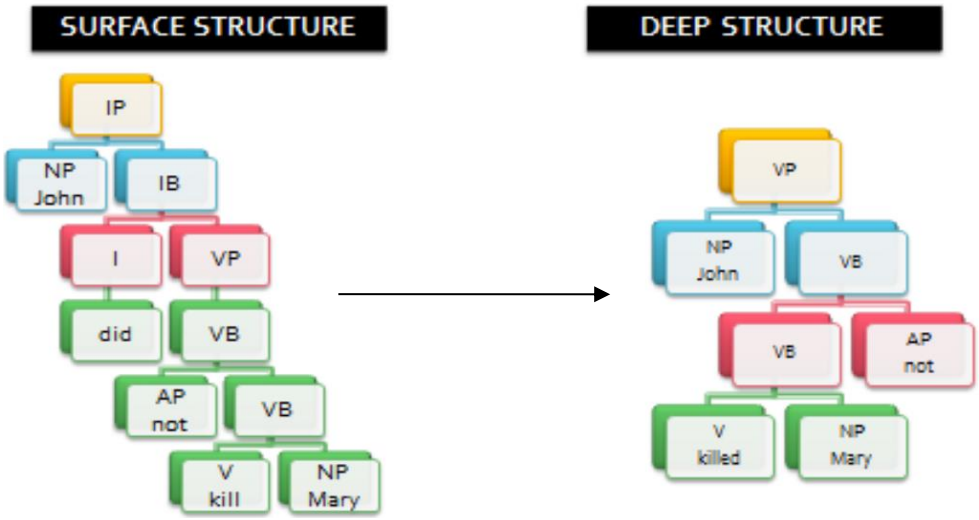


Figure 4.19: Transformation

C. Dearborization

UNL graph is a network rather than a tree. The deep syntactic structure obtained after transformation must be 'dearborized' in order to convert it into UNL. It is performed by rewriting X-Bar relations (*XP, XB*) as head-driven syntactic relations (*XS, XC, XA*). In dearborization, tree structures are converted into head-driven structures. These head driven structures are further converted into intermediate semantic relations like "*VS*", "*VC*", "*VA*" *etc*. In these relations, the first character of "*VS*", "*VC*", and "*VA*", *i.e.,* "*V*" indicates that the first argument is a verb, while second character "*S*", "*C*", and "*A*" indicates that the second argument of a relation is specifier, complement or an adjunct respectively. The network structure of example sentence given in (4.48) obtained after dearborization is shown in Figure 4.20.

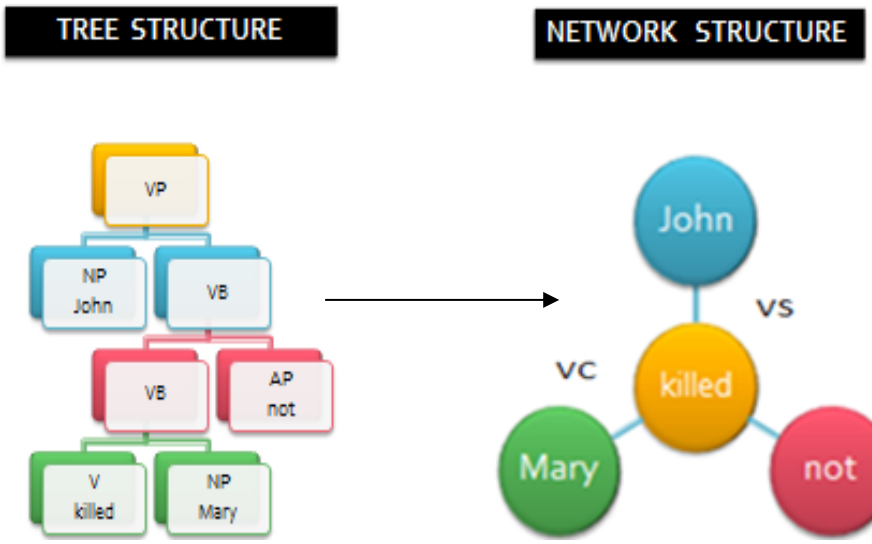


Figure 4.20: Dearborization

D. Interpretation

In interpretation, syntactic network obtained after dearborization is simply mapped to a semantic network that is obtained after analyzing the arguments of each relation as shown in Figure 4.21.

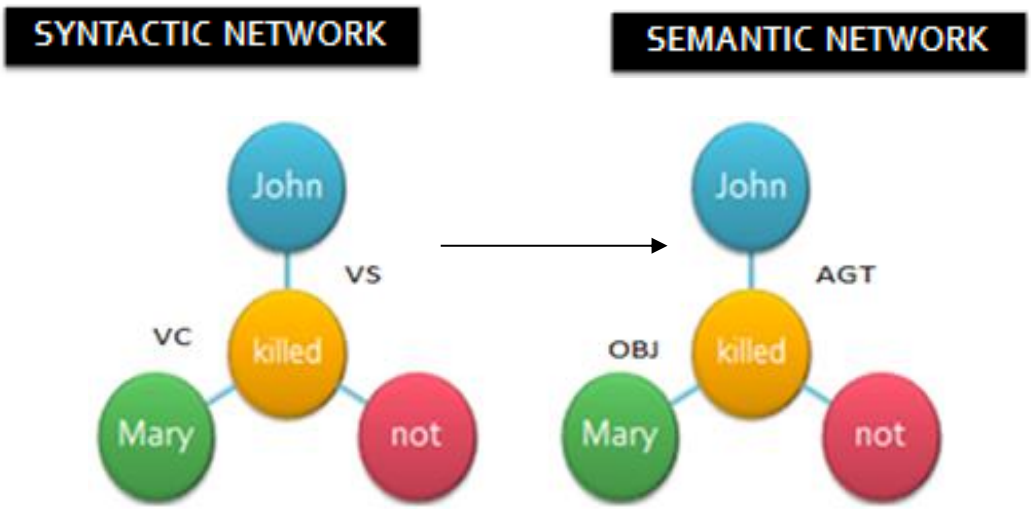


Figure 4.21: Interpretation

In the above example, node "*not*" assigns the attribute "*@not*" to the UW "*kill*". The node "*not*" does not form any relation with any node.

E. Rectification/ Post-processing

In post-processing, the resulting graph is adjusted according to UNL standards to remove redundancies and contradictions. For example, consider the rule given in 4.49.

(@pl,{@multal|@paucal|@all|@both}):=(-@pl); ...(4.49)

This rule eliminates the redundancy of "*@pl*". The idea of the plural is already being conveyed by "*@multal*", "*@paucal*", "*@all*", or "*@both*". Therefore, "*@pl*" is redundant and should be fixed. So, here "*@pl*" is removed with the help of post-processing rules.

4.10 Working of IAN/UNLization Module with an Example Sentence

The working of the proposed system has been explained by considering the example sentence taken from AESOP-A1 corpus. AESOP-A1 is the latest corpus provided by UNDL foundation. AESOP-A1 contains the famous story of "The Tortoise and the Hare" from Aesop's Fables. For UNLization, AESOP-A1 is manually converted into Punjabi language and uploaded to 'NL-Input' tab of IAN. The following subsections give a detailed explanation of UNLization of Punjabi natural language.

4.10.1 Normalization

Since the corpus is in a paragraph form hence, with the help of N-Grammar rule as given in (4.50), the paragraph is broken down to 13 sentences as shown in Table 4.18 as an input to IAN.

(%a,"|")(%b,^STAIL):=(%a)(STAIL)(%b) ...(4.50)

Here, '*%a'* refers to node '|' which indicates sentence end in Punjabi language, similar to the punctuation '.' in the English language. This N-Grammar adds the tag <STAIL> after every sentence end. The tag <SHEAD> is assigned automatically after <STAIL>. So, because of this N-Grammar rule, the paragraph is broken into sentences as given in Table 4.18.

Table 4.18: Sentences of AESOP-A1

Sr No.	Punjabi sentences of AESOP-A1
1.	ਖ਼ਰਗੋਸ਼ ਅਤੇ ਕੱਛੂਕੁੰਮਾ *ḵẖargōsh atē kacchūkummā* *"The Hare and the Tortoise"*
2.	ਖ਼ਰਗੋਸ਼ ਨੇ ਇਕ ਦਿਨ ਕੱਛੂਕੁੰਮੇ ਦੇ ਛੋਟੇ ਪੈਰਾਂ ਅਤੇ ਹੌਲੀ ਚਾਲ ਦਾ ਮਜ਼ਾਕ ਉਡਾਇਆ *ḵẖargōsh nē ik din kacchūkummē dē chōṭē pairāṃ atē haulī cāl dā mazāk uḍāiā* *"The Hare one day ridiculed the short feet and slow pace of the Tortoise"*

3.	ਕੱਛੂਕੁੰਮੇ ਨੇ ਜਵਾਬ ਦਿੱਤਾ *kacchūkummē nē javāb dittā* *The Tortoise replied*
4.	ਭਾਵੇਂ ਤੂੰ ਹਵਾ ਦੀ ਤਰ੍ਹਾਂ ਤੇਜ਼ ਹੈ,ਮੈਂ ਤੈਨੂੰ ਦੌੜ ਵਿਚ ਹਰਾ ਦੇਵਾਂਗਾ *bhāvēṃ tūṃ havā dī tarhāṃ tēz hai,maiṃ tainūṃ dauṛ vic harā dēvāṅgā* *"Though you be swift as the wind, I will beat you in a race"*
5.	ਖ਼ਰਗੋਸ਼ ਨੂੰ ਇਹ ਗੱਲ ਬਿਲਕੁਲ ਅਸੰਭਵ ਲੱਗੀ ਅਤੇ ਪ੍ਰਸਤਾਵ ਲਈ ਰਾਜੀ ਹੋ ਗਿਆ *ḵẖargōsh nūṃ ih gall bilkul asambhav laggī atē prastāv laī rājī hō giā* *"The Hare believed her assertion to be simply impossible and assented to the proposal"*
6.	ਉਹਨਾਂ ਨੇ ਮੰਨਿਆ ਕਿ ਲੂੰਬੜੀ ਪਥ ਚੁਣੇਗੀ ਅਤੇ ਟੀਚਾ ਮਿੱਥੇਗੀ *uhnāṃ nē manniā ki lūmbṛī path cuṇēgī atē ṭīcā mitthēgī* *"They agreed that the Fox should choose the course and fix the goal"*
7.	ਦੌੜ ਲਈ ਨਿਰਧਾਰਿਤ ਦਿਨ ਦੌਵਾਂ ਨੇ ਇਕੱਠੇ ਸ਼ੁਰੂ ਕੀਤਾ *dauṛ laī nirdhārit din dauvāṃ nē ikṭṭhē shurū kītā* *"On the day appointed for the race the two started together"*
8.	ਕੱਛੂਕੁੰਮਾ ਰੁਕਿਆ ਨਹੀਂ *kacchūkummā rukiā nahīṃ* *"The Tortoise did not stop"*
9.	ਉਹ ਹੌਲੀ ਪਰ ਲਗਾਤਾਰ ਪਥ ਦੇ ਅਖੀਰ ਤਕ ਸਿੱਧਾ ਜਾਂਦਾ ਗਿਆ *uh haulī par lagātār path dē akhīr tak siddhā jāndā giā* *"She went on with a slow but steady pace straight to the end of the course"*
10.	ਖ਼ਰਗੋਸ਼ ਰਸਤੇ ਵਿਚ ਥੱਲੇ ਲੰਮਾ ਪੈ ਗਿਆ ਅਤੇ ਇਕ ਰੁੱਖ ਹੇਠ ਝਪਕੀ ਲੈ ਗਿਆ *ḵẖargōsh rastē vic thallē lammā pai giā atē ik rukkh hēṭh jhapkī lai giā* *"The Hare laid down by the wayside and took a nap under a tree"*
11.	ਆਖਰ, ਉਹ ਉੱਠਿਆ ਅਤੇ ਦੌੜਿਆ ਜਿੰਨੀ ਤੇਜ਼ ਉਹ ਦੌੜ ਸਕਦਾ ਸੀ, ਪਰ ਇਹ ਬਹੁਤ ਦੇਰ ਸੀ *ākhar, uh uṭṭhiā atē dauṛiā jinnī tēz uh dauṛ sakdā sī, par ih bahut dēr sī* *"At last, he woke up and ran as fast as he could, but it was too late"*
12.	ਕੱਛੂਕੁੰਮਾ ਪਹਿਲਾਂ ਹੀ ਦੌੜ ਨੂੰ ਜਿੱਤ ਚੁੱਕਿਆ ਸੀ *kacchūkummā pahilāṃ hī dauṛ nūṃ jitt cukkiā sī* *"The Tortoise had already won the race"*
13.	ਹੌਲੀ ਪਰ ਲਗਾਤਾਰ ਤਰੱਕੀ ਦੌੜ ਨੂੰ ਜਿੱਤਦੀ ਹੈ *haulī par lagātār tarkkī dauṛ nūṃ jittdī hai* *"Slow but steady progress wins the race"*

Out of the sentences given in Table 4.18, UNLization of Punjabi natural language is illustrated by considering the example sentence given in (4.51).

ਖ਼ਰਗੋਸ਼ ਨੇ ਇਕ ਦਿਨ ਕੱਛੂਕੁੰਮੇ ਦੇ ਛੋਟੇ ਪੈਰਾਂ ਅਤੇ ਹੌਲੀ ਚਾਲ ਦਾ ਮਜ਼ਾਕ ਉਡਾਇਆ ...(4.51)

ḵẖargōsh nē ik din kacchūkummē dē chōṭē pairāṃ atē haulī cāl dā mazāk uḍāiā

"The hare one day ridiculed the short feet and slow pace of the tortoise".

4.10.2 Tokenization

During tokenization of example sentence given in (4.51) with IAN tool, twenty two lexical items are identified as given in (4.52).

[ਖ਼ਰਗੋਸ਼]{}"hare"(LEX=N,POS=NOU,GEN=MCL ,NUM=SNG)<pan,0,0>;

[ਨੇ]{-1}""(LEX=P,POS=PPS,rel=agt)<pan,0,0>;

[ਇਕ ਦਿਨ]{}"one day"(LEX=N,POS=NOU,rel=tim)<pan,0,0>;

[ਕੱਛੂਕੁੰਮੇ]{}"tortoise"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>;

[ਦੇ]{}""(LEX=P,POS=PPS,rel=mod)<pan,0,0>;

[ਛੋਟੇ]{}"short"(LEX=J,POS=ADJ)<pan,0,0>;

[ਪੈਰਾਂ]{}"foot"(LEX=N,POS=NOU,NUM=PLR)<pan,0,0>;

[ਅਤੇ]{}"and"(LEX=C,POS=COO,rel=and)<pan,0,0>;

[ਹੌਲੀ]{}"slow"(LEX=J,POS=ADJ,NUM=SNG)<pan,0,0>;

[ਚਾਲ]{}"pace"(LEX=N,POS=NOU,NUM=SNG)<pan,0,0>;

[ਦਾ]{}""(LEX=D,POS=ART,att=@def)<pan,0,0>;

[ਮਜ਼ਾਕ ਉਡਾਇਆ]{}"ridicule"(LEX=V,POS=VER,att=@past,NUM=SNG)<pan,0,0>;

Eleven blank spaces are also identified as:-

[]{}""(BLK)<pan,0,0>; ...(4.52)

4.10.3 Parsing

After parsing, the example sentence given in (4.51) is converted to tree structure as shown in Figure 4.22.

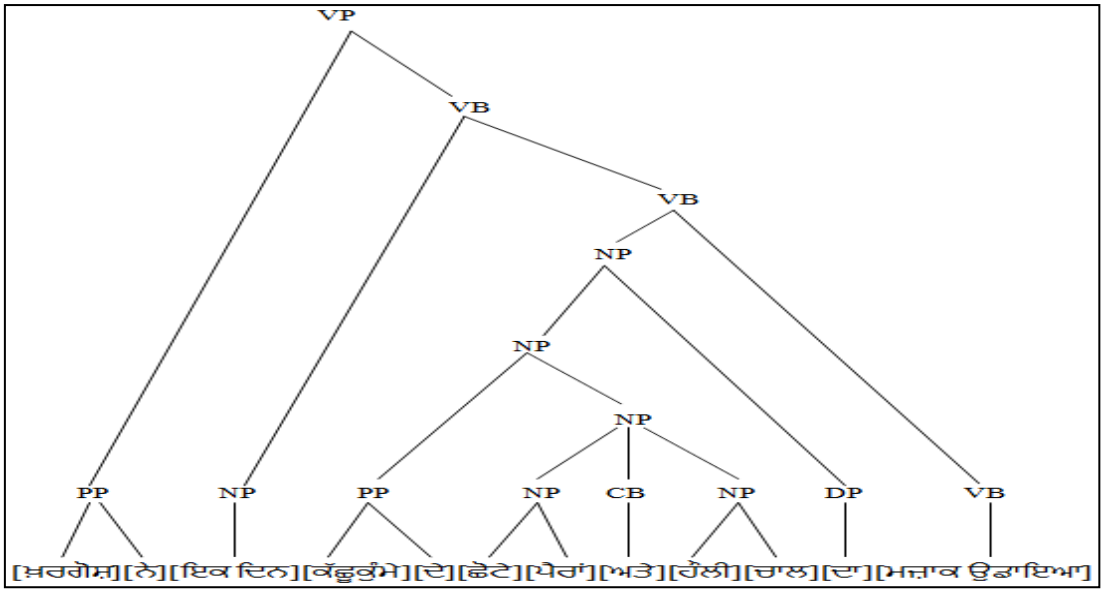


Figure 4.22: Tree Structure for Example Sentence Given in (4.51)

The UNL generated after parsing phase is given in (4.53).

NP:01(foot.@pl;short)

NP:02(pace;slow)

and:03(02;01)

NP:04(03;tortoise.@def)

VB:05(ridicule.@past;04)

VB:06(05;one day)

VP(06;hare.@def) …(4.53)

4.10.4 Transformation

As described in section 4.10.1, in transformation phase, the surface tree structure is converted into a modified tree so as to represent its deep syntactic structure. However, in the given example there is no need to convert the surface tree structure as given in Figure 4.22 because in this surface tree structure, its deep syntactic structure is already exposed.

4.10.5 Dearborization

In dearborization phase, the example sentence given in (4.51) is converted to network structure.

The UNL generated after dearborization phase is given in (4.54).

NA:01(foot.@pl,short)

NA:02(pace,slow)

and:03(02:01)

VA(ridicule.@past,one day)

VS(ridicule.@past,hare.@def)

NA:04(03,tortoise.@def)

VC(ridicule.@def,04) ...(4.54)

4.10.6 Interpretation

In Interpretation phase, the example sentence given in (4.51) is converted to semantic network.

The UNL generated after interpretation is given in (4.55).

agt(ridicule.@past,hare.@def)

tim(ridicule.@past,one day)

obj(ridicule.@past,:04)

mod:04(:03,tortoise.@def)

and:03(:02,:01)

mod:02(pace,slow)

mod:01(foot.@pl,short) ...(4.55)

The UNL generated after the interpretation phase does not require any rectification or post-processing because there are no contradictions and redundancies. The final UNL graph of the example sentence given in (4.52) is shown in Figure 4.23.

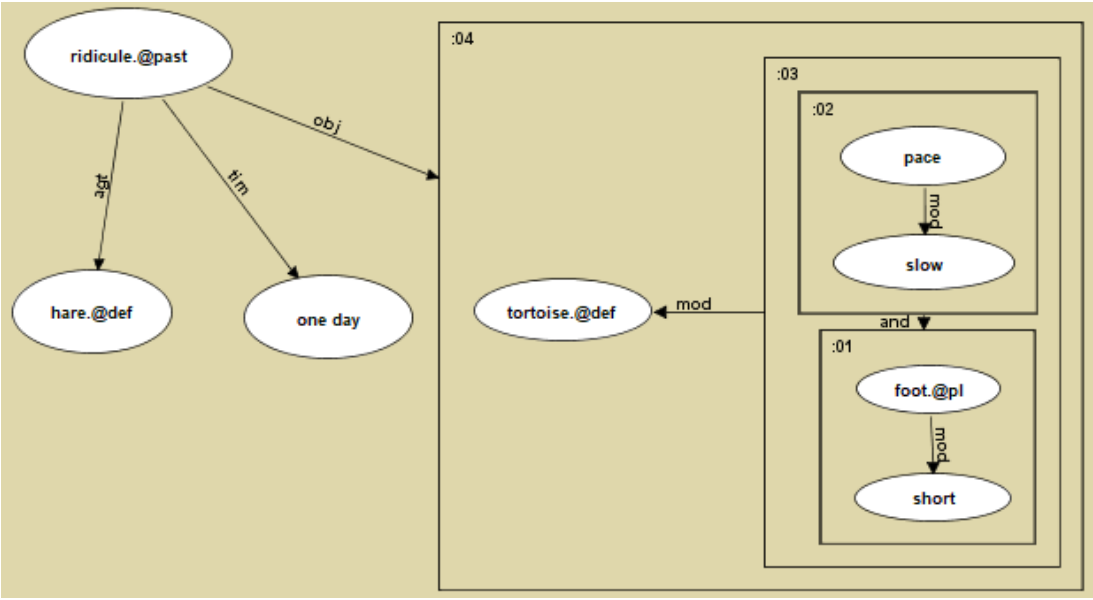


Figure 4.23: UNL Graph of Example Sentence Given in (4.51)

As discussed previously in this chapter, the purpose of the IAN/UNLization module is to convert natural language sentence (*i.e.*, the question asked by the user through

developed QAS's interface) to UNL. The analysis module invokes UNLization/IAN module of the source natural language (Punjabi in this case). In order to UNLize any given corpus or question asked by the user, IAN uses the natural language dictionary, normalization rules (NRules), transformation rules (TRules) and disambiguation rules (DRules) of the source natural language. These resources are created according to UNL specifications [153]. Dictionary, NRules, TRules, and DRules are created by the computational linguists of the respective natural language and stored at UNL web [160]. IAN converts the natural language corpus/sentences into UNL, which on being invoked by the analysis module of developed question answering system returns its UNL.

Similar to IAN, with the help of EUGENE (dEep-to-sUrface GENErator) tool UNL can be converted to any natural language with the help of generation dictionary, TRules, and DRules. This process is called NLization or Generation. Details of EUGENE have been covered in Chapter 6 of this thesis. The way analysis module of the developed question answering system invokes IAN, its generation module invokes EUGENE/NLization module to convert the answer into the target language (Punjabi in this case).

Section 4.11 illustrates that how IAN and EUGENE are configured and invoked by the analysis and generation modules of the developed question answering system.

4.11 Configuring and Invoking IAN (UNLization Module) and EUGENE (NLization Module) of the Proposed QAS

The analysis module and generation module of the developed question answering system invoke IAN and EUGENE module of the source and target natural language for UNLization and NLization respectively. Before IAN and EUGENE can be invoked, certain settings and additions need to be done which are a kind of prerequisites for this.

4.11.1 Prerequisites of the Proposed System

UNLization and NLization of a natural language can be done using the proposed system if its IAN and EUGENE modules are present. The owner of that module should have configured the web service from its account on UNL web [160]. The configuration of web service from an account on UNL dev (www.unlweb.net/user/index.php) has been explained in the following steps with an

example of Punjabi language.

Step 1: Sign in to the existing UNL web account from the login page as shown in Figure 4.24 [160].

Figure 4.24: Step 1 to Configure Web Service From UNL Web

Step 2: After signing in, the user needs to navigate to UNL dev as highlighted in Figure 4.25.

Figure 4.25: Home Page of Logged in User Containing UNL Dev Link

Step 3: After navigating to UNL dev, the user needs to navigate to Web service configuration screen from WS EDITOR as shown in Figure 4.26.

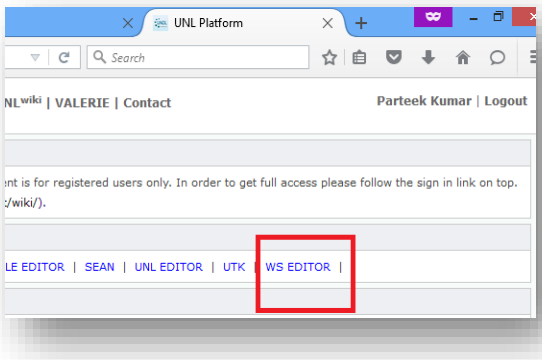


Figure 4.26: UNL Dev Page of Logged in User Containing Web Service Editor Link

Step 4: After step 3, user will be navigated to the Web service editor as shown below in Figure 4.27 and after clicking on UNL WS Configuration link, the user will be redirected to its WS Client configuration home page as shown in Figure 4.27.

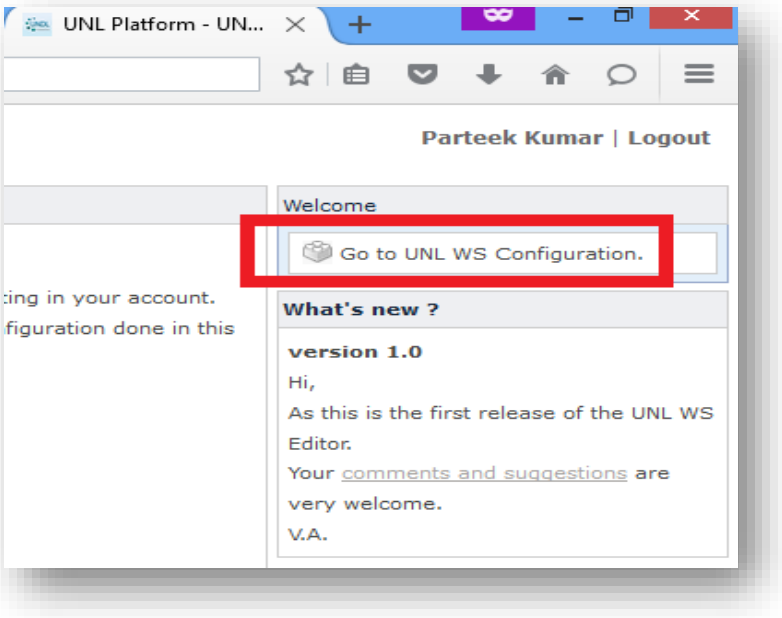


Figure 4.27: Web Service Editor

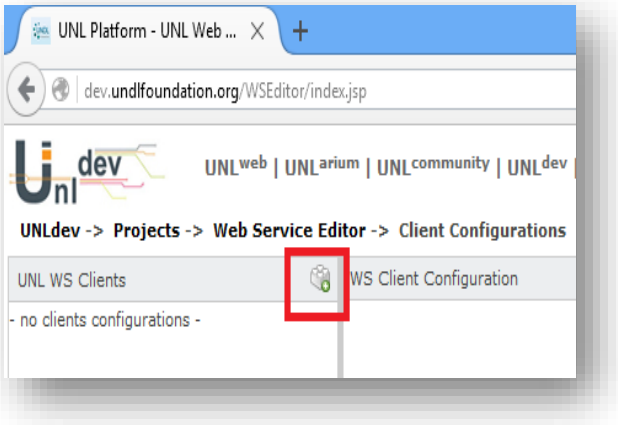


Figure 4.28: Web Service Client Configuration Page for Logged in User

Step 5: Add a WS Client from the button as highlighted in Figure 4.28 and when prompted name it as 'test'. After that, the 'test' client configuration will be created for the logged in user as shown in Figure 4.29. The four root elements correspond to the available engines (IAN, EUGENE, SEAN, and NORMA) and are being created automatically when a client is created. Each engine will need to be configured separately. The lists of available languages are shown under each engine. The list consists of language names followed by a 3-char ISO code assigned.

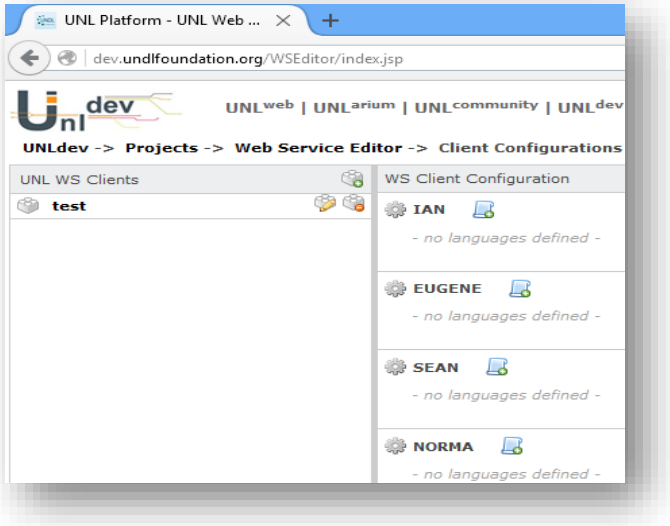


Figure 4.29: 'test' Client Configuration After Step 4

Step 6: Now, the user has to configure IAN and EUGENE resources for UNLization and NLization. New language configurations for each engine can be added by a button next to the engine title. The languages items in the tree are expandable. When expanded, categories of resources are shown depending on the engine. Analysis engines like IAN have 4 types of resources predefined (Dictionary, N-Rules, D-Rules, and T-Rules). Generation engines like EUGENE do not have N-Rules type. A button next to each type will open a dialog with the list of all available resources of that type owned by the user or shared with him by other users. Owned and shared resources will be marked with different icons in the tree when the selection is confirmed by clicking of save button. The sample configuration has been shown in Figure 4.30 for the Punjabi language.

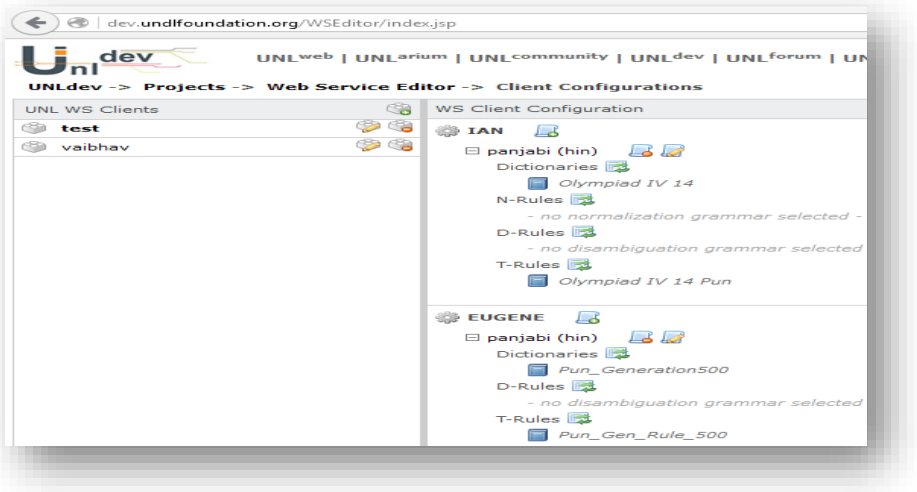


Figure 4.30: Web Service Configuration of IAN and EUGENE for Punjabi Language

Now at this point of time, web service for the Punjabi language has been successfully configured. Even if now user loads different resources for IAN and EUGENE in its UNL web account (not in web service), that won't change web service configuration of IAN and EUGENE.

Since, the aim of the proposed cross domain client application is to support multilingualism, it is expected for the registered users to share their user name, web service client name, and the 3 character ISO codes assigned to IAN and EUGENE configuration. For example, in the above configuration user name is 'parteek', the web service client name is 'test', 3 character ISO code for IAN is 'hin', and 3 character ISO code for EUGENE is 'hin'. Invoking of these modules with the help of this configuration information is described in section 4.11.2.

4.11.2 Invoking of IAN/UNLization and EUGENE/NLization Module from the Proposed Question Answering System

Figure 4.31 shows the proposed queation answering system's client configuration snapshot. The users who want their language to be UNLize or NLize by other researchers around the world can upload their user name, web service client name, and the 3 character ISO codes assigned to IAN and EUGENE configuration from 'SHARE' section. All the shared language names will be available in the drop-down menu and client can select any of those shared languages as shown in Figure 4.31.

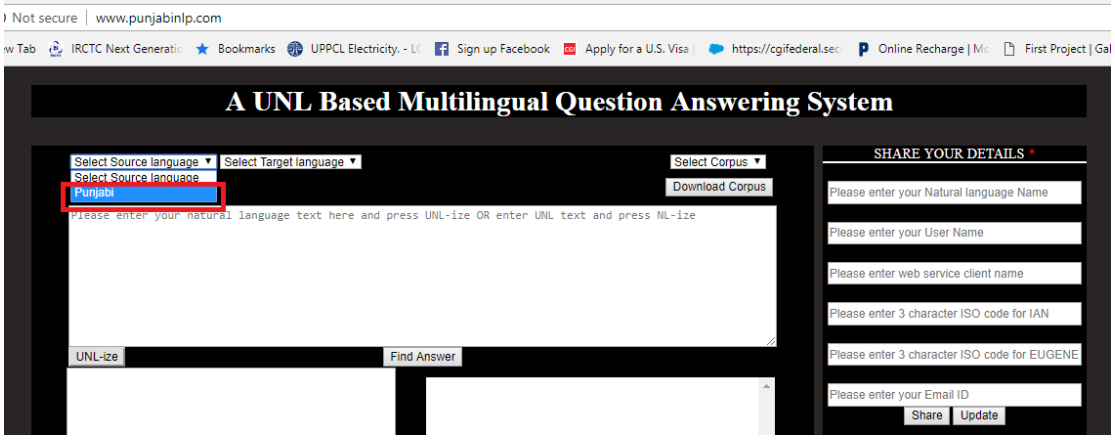


Figure 4.31: Snapshot of the Proposed Client Configuration

Let's say the user selects Agro-Explorer corpus from the drop down and selects 'Punjabi' as the natural language from the language drop-down options. Consider example sentence (4.56) taken from Agro-Explorer corpus which user types in the given text area as shown in Figure 4.32.

ਆਧੁਨਿਕ ਖੇਤੀਬਾੜੀ ਕਿਹੜੀ ਚੀਜ਼ ਉੱਪਰ ਨਿਰਭਰ ਹੈ?

Ādhunika khētībāṛī kihaṛī cīza upara nirabhara hai?

"Modern agriculture depends on what?" ...(4.56)

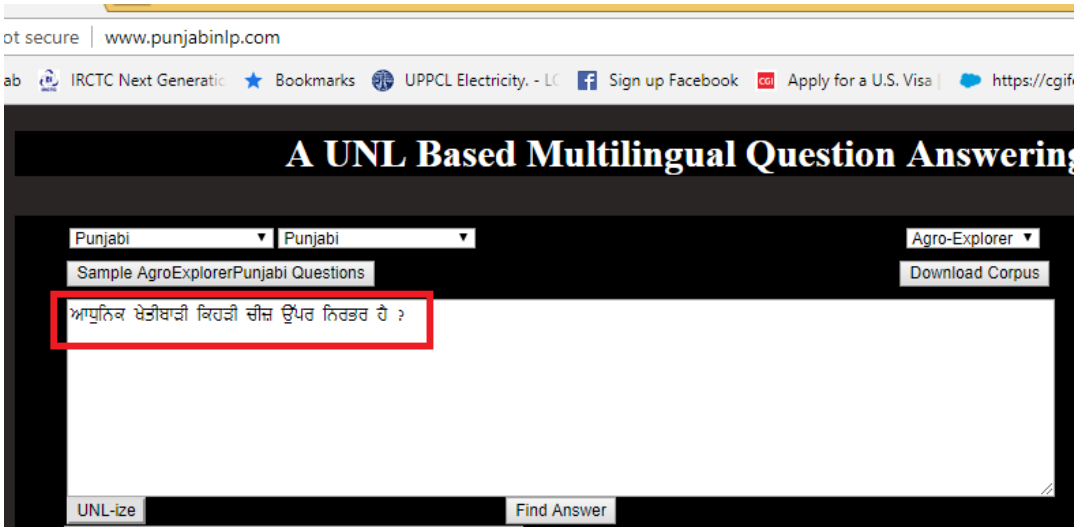


Figure 4.32: Snapshot of the Question Asked by the User

When the user clicks on the 'Find Answer' or 'UNL-ize' button, the analysis module for 'Punjabi' natural language is called and we get the UNL of the input question as shown in (a) of Figure 4.33.

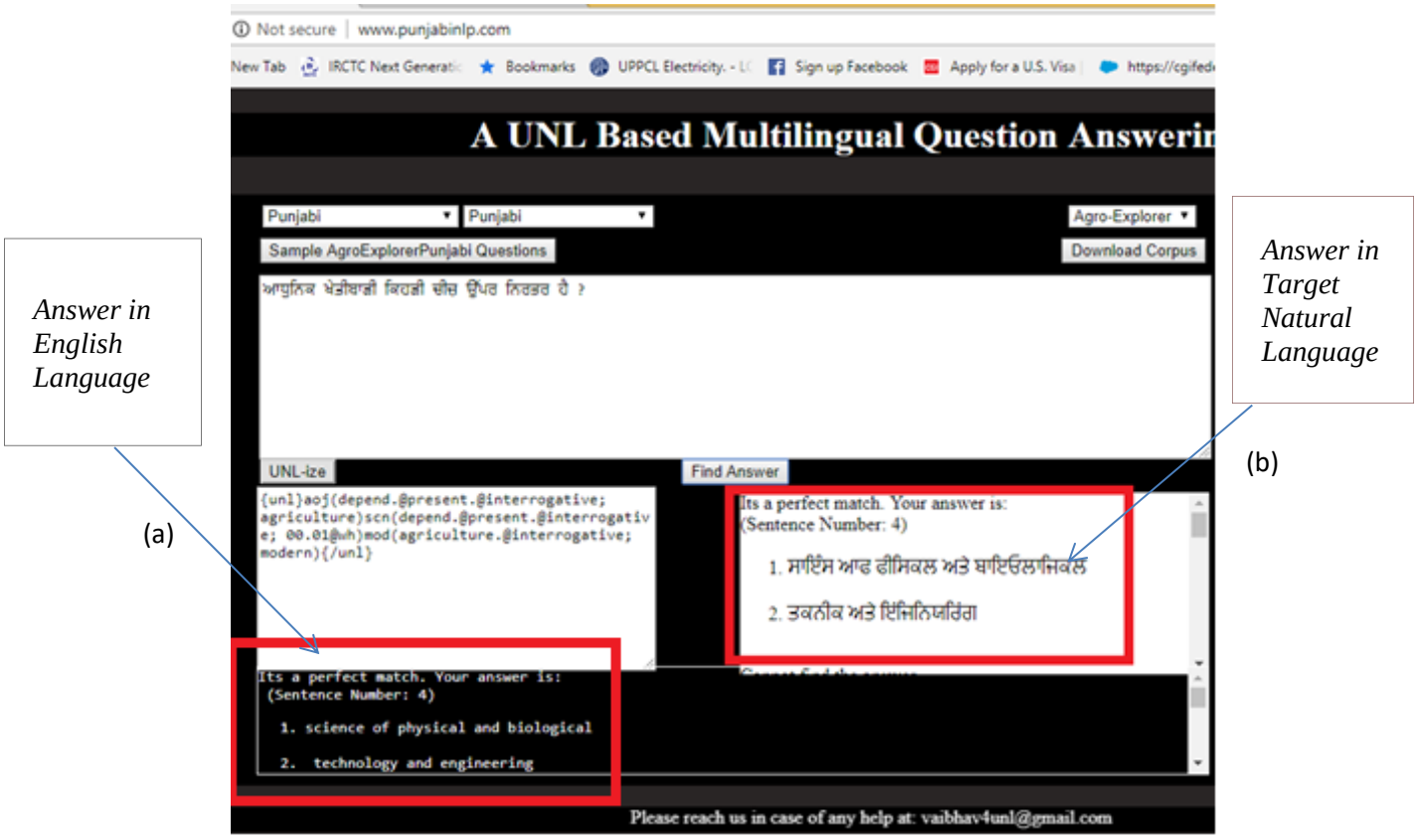


Figure 4.33: UNL of the Input Question

On a similar basis, when the question answering system finds the answer to the asked question, then EUGENE module of the language selected from the target language drop-down is called automatically and the user can see this answer in the selected language as shown in (b) of Figure 4.33. The EUGENE/NLization and generation modules have been discussed in detail in Chapter 6.

4.12 Evaluation Metrics and Results of UNLization/ IAN Module

The correctness of UNLization and NLization module of the selected language depends on their F-Measure. The F-Measure/F1-score is calculated with the help of an online tool developed by the UNDL foundation available at UNL-arium [30]. Two parameters required for the calculation of F-Measure are Precision and Recall. F-Measure is calculated by the formula given in (4.57) [30].

$$F - Measure = 2 * \{(Precision * Recall)/(Precision + Recall)\} \quad \ldots(4.57)$$

Precision is the number of correct results divided by the number of all returned results. The Recall is the number of correct results divided by the number of results that should have been returned. If the output is in the form of graph and is made only of UWs, then result is considered to be "returned". If the discrepancy of UWs and relations between the actual and the expected output; and the overall discrepancy is less than 0.3, then the result is said to be "correct".

4.12.1 Testing Corpus

The UNLization module has been tested on UC-A1, UGO-A1, AESOP-A1, CORPUS 500, and UC-A2 corpora. These corpora are provided by the UNDL foundation to all the participant languages. These corpora covers all the major part-of-speeches. Table 4.19 depicts the categorization of these corpora.

Table 4.19: Categorization of UC-A1, UGO-A1, AESOP-A1, CORPUS 500, and UC-A2

Sr. No	Sentence Type	Number of sentences in UC-A1	Number of sentences in UGO-A1	Number of sentences in AESOP-A1	Number of sentences in CORPUS 500	Number of sentences in UC-A2
1.	Sentence structures	20	40	13	40	10
2.	Temporary words	5	14	-	10	-
3.	Numbers and Ordinals	10	25	-	50	-
4.	Nouns	15	35	-	60	40
5.	Adjectives	7	15	-	50	40
6.	Determiners	9	20	-	40	30
7.	Prepositions	6	25	-	50	40
8.	Pronouns	5	20	-	50	40
9.	Time	5	12	-	50	30
10.	Verb	9	24	-	50	40
11.	Conjunctions	9	20	-	50	30
TOTAL		**100**	**250**	**13**	**500**	**300**

AESOP-A1 contains the famous story of "*The Tortoise and the Hare*" from Aesop's Fables. For testing these corpora, the Punjabi natural language corpus was uploaded on UNL Web and UNLization of the entire corpus was done in one go using IAN (separately for each corpus). The output UNL was copied from IAN console and saved in UTF-8 format. This is the actual output file. UNL web also provides the expected output file (in UNL format). These actual and expected UNL files are then uploaded at UNL-arium and the F-Measure is then calculated by the system.

UNLization module of Punjabi language has 24 NRules, 48 DRules, 982 TRules and 771 Dictionary entries. Table 4.20 shows the details of IAN module for UNLization of Punjabi language.

Table 4.20: Details of IAN/UNLization Module for UNLization of Punjabi Language

UNLization Assets	Count/ Value
Dictionary entries	771
NRules	24
DRules	48
TRules	982

4.12.2 Results of IAN/UNLization Module

For Punjabi language UNLization module, the values of Precision, Recall, number of processed, returned and correct sentences for UC-A1, UGO-A1, AESOP-A1, CORPUS 500, and UC-A2 are given in Table 4.21.

Table 4.21: Testing details of UC-A1, UGO-A1, and AESOP-A1

S. No	Parameters	UC-A1 Value	UGO-A1 Value	AESOP-A1 Value	CORPUS 500 Value	UC-A2 Value
1.	Sentences processed	100	250	13	500	300
2.	Sentences returned	100	250	13	471	287
3.	Sentences correct	97	247	13	401	264
4.	Precision	0.970	0.988	1.00	0.851	0.880
5.	Recall	0.970	0.992	1.00	0.802	0.920
6.	F-Measure	0.970	0.990	1.00	0.826	0.900

4.13 Errors/Discrepancies in the IAN/UNLization Module

As explained above, the accuracy of the IAN/UNLization module is calculated using an online tool (provided by UNDL foundation) available at UNL-arium. The F-Measure of a corpus calculated by the tool depends upon the following discrepancies.

- Discrepancy of relations
- Discrepancy of UWs
- Overall Discrepancy

The meaning of these discrepancies and formulae to calculate it are explained in detail using example sentences in following sub sections.

4.13.1 Errors due to Discrepancy of Relations

As the name suggests, these are the errors which get introduced when there is a discrepancy in the output UNL's relation. These discrepancies are calculated by the formula given in (4.58).

$$Discrepancy\ of\ relations = \frac{exceeding\_rel + missing\_rel}{total\_rel} \quad ...(4.58)$$

Where, 'exceeding_rel' is the number of UNL relations presents in the actual output but absent from the expected output, 'missing_rel' is the number of UNL relations absent from the actual output but present in the expected output and 'total_rel' is the sum of the total number of relations in the actual output and in the expected output.

Consider an example sentence (4.59) taken from AESOP-A1 corpus.

ਖ਼ਰਗੋਸ਼ ਨੇ ਇਕ ਦਿਨ ਕੱਛੂਕੁੰਮੇ ਦੇ ਛੋਟੇ ਪੈਰਾਂ ਅਤੇ ਹੌਲੀ ਚਾਲ ਦਾ ਮਜ਼ਾਕ ਉਡਾਇਆ

ḵẖargōsh nē ik din kacchūkummē dē chōṭē pairāṃ atē haulī cāl dā mazāk uḍāiā

"The hare one day ridiculed the short feet and slow pace of the tortoise". ...(4.59)

The expected UNL of this example sentence (4.59) is specified in (4.60).

{unl}
agt(ridicule.@past,hare.@def)
tim(ridicule.@past,one day)
obj(ridicule.@past,:04)
mod:04(:03,tortoise.@def)
and:03(02,01)
mod:02(pace,slow)
mod:01(foot.@pl,short)
{/unl} ...(4.60)

Let's say that the actual output from IAN is:

{unl}
agt(ridicule.@past,hare)
cnt(ridicule.@present,one day.@multal)
mod:04(:03,tortoise)
aoj:02(ridicule,slow)
aoj:01(foot,short)
{/unl} ...(4.61)

Here,

- exceeding_rel = 3, which are "*cnt(ridicule.@present,one day.@multal)*", "*aoj:02(ridicule,slow)*", and *"aoj:01(foot,short)"* since these are present in actual UNL but not in expected UNL.

- missing_rel = 5, which are "*tim(ridicule.@past,one day)*", "*obj(ridicule.@past,:04)", "and:03(02,01)",* "*mod:02(pace,slow)*", and "*mod:01(foot.@pl,short)"* since these are present in expected UNL but not in actual UNL.
- total_rel = 12, since 7 relations are in expected UNL and 5 relations are in actual UNL.

Now, after substituting these values in (4.58), discrepancy of relations comes out to be as given below.

Discrepancy of Relations = (3 + 5) / 12 = 0.67.

4.13.2 Errors due to Discrepancy of Universal Words

These are the errors which get introduced when there is a discrepancy in the output UNL's Universal Words (UWs). These discrepancies are calculated by the formula given in (4.62).

$$Discrepancy\ of\ UWs = \frac{exceeding\_UW + missing\_UW}{total\_UW} \quad ...(4.62)$$

Where, 'exceeding_UW' is the numbers of UWs present in the actual output but absent from the expected output, 'missing_UW' is the number of UWs absent from the actual output but present in the expected output and 'total_UW' is the sum of the total number of universal words in the actual output and in the expected output.

For the given example sentence (4.59) having its expected UNL output given in (4.60) and actual UNL output given in (4.61):

- exceeding_UW = 0 because there are no UWs which are present in actual UNL but not present in expected UNL.
- missing_UW = 1, which is "*pace*" since it present in expected UNL but not in actual UNL.
- total_UW = 19, since 10 UWs are in expected UNL (scopes are not considered) and 9 UWs (scopes are not considered) are in actual UNL.

Now, after substituting these values in (4.58), discrepancy of UWs comes out to be as given below.

Discrepancy of UWs = (0 + 1) / 19 = 0.05.

4.13.3 Overall Discrepancy

These are the errors which get introduced due to discrepancy in the output UNL's attributes, discrepancy of UWs, and discrepancy of relations. These discrepancies are calculated by the formula given in (4.63).

$$Overall \quad Discrepancy = \frac{3 \times (exceeding\_rel + missing\_rel) + 2 \times (exceeding\_UW + missing\_UW) + (exceeding\_attr + missing\_attr)}{3 \times total\_rel + 2 \times total\_UW + total\_attr} \quad \ldots(4.63)$$

Where, 'exceeding_attr' is the numbers of attributes present in the actual output but absent from the expected output, 'missing_attr' is the number of attributes absent from the actual output but present in the expected output and 'total_attr' is the sum of the total number of attributes in the actual output and in the expected output.

For the given example sentence (4.59) having its expected UNL output given in (4.60) and its actual UNL output given in (4.61):

- exceeding_attr = 2, which are "*@present"* and *"@multal*" since these are present in actual UNL but not in expected UNL.
- missing_attr = 5, which are "*@def*", "*@past", "@past",* "*@def*", and "*@pl"* since these are present in expected UNL but not in actual UNL.
- total_attr = 9, since 6 attributes are in expected UNL and 3 attributes are in actual UNL.

Now, after substituting these values in (4.58), overall discrepancy comes out to be as given below.

$$Overall\ Discrepancy = \frac{3 * (3 + 5) + 2 * (0 + 1) + (2 + 5)}{3 * 12 + 2 * 19 + 9} = 0.39$$

4.14 Achievements and Contributions of the Developed Punjabi IAN/UNLization Module

UNDL foundation had conducted a series of Olympiads over the years. These Olympiads for UNLization and NLization are held so as to find the best IAN and EUGENE modules for the participating languages across the globe and check their respective accuracies. Till date more than 33 languages are part of this UNL programme. The UNLization module for Punjabi language had been submitted for UNL Olympiad II, III, and IV conducted by UNDL foundation in July 2013, March 2014, and November 2014 for UC-A1, UGO-A1, and AESOP-A1 respectively [129][150][49].

The language selected for the proposed question answering system, *i.e.*, Punjabi, had been selected in top 10 (Based upon the F-Measures) UNLization grammars for Olympiad II, while it was selected in top 5 best grammars for Olympiad III and Olympiad IV. F-Measure of Punjabi IAN module and other best selected languages for II, III, and IV Olympiad is shown in Figures 4.34, 4.35 and 4.36 respectively. The highlighted entries in Figures 4.34, 4.32, and 4.33 refer to Punjabi language results.

10 Best UNLization Grammars

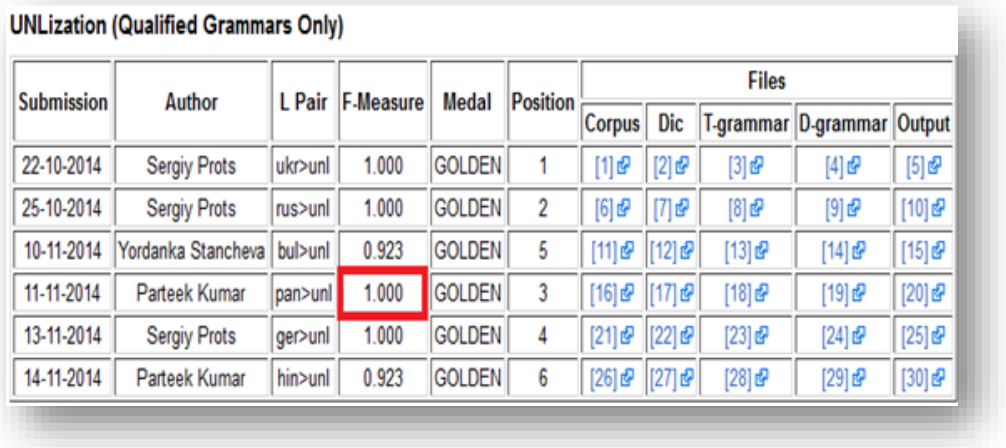
ABSOLUTE POSITION	ID	LPAIR	FMEASURE	SUBMITTED	AUTHOR
1	29	slv>unl	1.000	05/04/2013 07:28	Grega Milharcic
2	32	ger>unl	1.000	16/04/2013 15:26	Sergiy Prots
3	36	srp>unl	1.000	28/04/2013 11:36	Sergiy Prots
4	37	hrv>unl	1.000	29/04/2013 14:04	Sergiy Prots
5	43	swe>unl	1.000	24/05/2013 00:19	Sergiy Prots
6	46	lat>unl	1.000	05/06/2013 12:55	Sergiy Prots
7	51	tur>unl	1.000	23/06/2013 05:21	Sergiy Prots
8	52	afr>unl	1.000	24/06/2013 21:32	Grega Milharcic
9	80	pan>unl	0.970	01/07/2013 11:09	Parteek Kumar
10	58	khm>unl	0.950	28/06/2013 09:52	Sokphyrum Kim

Figure 4.34: F-Measure for Top 10 Best UNLization Selected Languages in Olympiad II [129]

UNLization

General Position	Language Pair	Author	F-Measure	Submission Date	Medal	Files			
						Dictionary	T-Grammar	D-Grammar	Output
1	slv>unl	Grega Milharcic	1.000	07/03/2014	GOLD	[1]	[2]	[3]	[4]
2	pan>unl	Parteek Kumar	0.990	29/03/2014	GOLD	[5]	[6]	[7]	[8]
3	bul>unl	Yordanka Stancheva	0.976	21/03/2014	GOLD	[9]	[10]	[11]	[12]
4	ukr>unl	Sergiy Prots	0.946	25/03/2014	GOLD	[13]	[14]	[15]	[16]
5	rus>unl	Sergiy Prots	0.942	23/03/2014	GOLD	[17]	[18]	[19]	[20]

Figure 4.35: F-Measure for Top 5 Best UNLization Selected Languages in Olympiad III [150]



UNLization (Qualified Grammars Only)

Submission	Author	L Pair	F-Measure	Medal	Position	Files				
						Corpus	Dic	T-grammar	D-grammar	Output
22-10-2014	Sergiy Prots	ukr>unl	1.000	GOLDEN	1	[1]	[2]	[3]	[4]	[5]
25-10-2014	Sergiy Prots	rus>unl	1.000	GOLDEN	2	[6]	[7]	[8]	[9]	[10]
10-11-2014	Yordanka Stancheva	bul>unl	0.923	GOLDEN	5	[11]	[12]	[13]	[14]	[15]
11-11-2014	Parteek Kumar	pan>unl	1.000	GOLDEN	3	[16]	[17]	[18]	[19]	[20]
13-11-2014	Sergiy Prots	ger>unl	1.000	GOLDEN	4	[21]	[22]	[23]	[24]	[25]
14-11-2014	Parteek Kumar	hin>unl	0.923	GOLDEN	6	[26]	[27]	[28]	[29]	[30]

Figure 4.36: F-Measure for Top 6 Best UNLization Selected Languages in Olympiad IV [49]

4.15 Details of IAN/UNLization Module for Proposed Question Answering System

The proposed system was tested on total of 400 questions which ranges over different question types like “Yes-No”, “What”, “Who”, “Where”, “When”, “Why”, “Which”, “How”, and “What”. The complete testing details like corpus used, metrics used for evaluating developed question answering system *etc.* have been covered in Chapter 7 of this thesis. Out of these 400 questions, total number of questions of each type has been given in Table 4.22.

Table 4.22: Types of Questions

Sr No.	**Question Type**	**Count**
1.	Yes-No	70
2	What	60
3.	Who	50
4.	Where	40
5.	When	40
6.	Why	40
7.	Which	40
8.	How	30
9.	What	30
Total	-	**400**

These questions were asked in Punjabi language through the interface of the proposed QAS as depicted in Figure 4.32. UNL of the corpus is already available (which forms the UNL repository), but in order to convert these questions to UNL, some new

dictionary entries, TRules, and DRules were created and merged with the existing IAN module. For the developed question answering system, Table 4.23 gives the details of total UNLization/IAN assets after merging with the existing IAN module.

Table 4.23: Details of IAN/UNLization Module of Punjabi Language for Proposed Question Answering System

UNLization Assets	**Count/Value**
Dictionary entries	1798
NRules	24
DRules	50
TRules	1259

Chapter Summary

In this chapter, the detailed information of analysis, UNLization/IAN module for the Punjabi language has been documented. This chapter explains the framework of IAN tool which is used for UNLization. This chapter gives the idea of phases of the UNLization process. Each of these phases has been explained in this chapter with the help of example sentences. The documentation regarding UNLization artifacts like Normalization, TRules, DRules, and Analysis Grammar *etc.* has been done in this chapter. This chapter introduces the X-Bar theory and its need in UNLization. The method of UNLization/ Transformation using this X-Bar theory has been illustrated in this chapter. The working of the developed IAN/UNLization module using IAN has been explained using example sentences. In this chapter a brief introduction about EUGENE is also given so that the configuration and invoking of IAN/UNLization and EUGENE/NLization modules should be clear to the reader.

This chapter highlights the use of UNLization/IAN and NLization/EUGENE modules from the developed question answering system perspective and gives details about how analysis and generation modules of the proposed QAS can be used to invoke UNLization/IAN and NLization/EUGENE modules respectively. It gives details about the prerequisites and other necessary steps for configuration.

After explaining about IAN, analysis and UNLization/IAN modules, the evaluation metrics and results of the UNLization/IAN module of Punjabi language have been described in this chapter. The same section also gives details of the corpora that are used for testing the UNLization/IAN module. This chapter highlights the types of errors which exist in the UNLization/IAN modules due to which F-Measure becomes less than 1. The details of method used to calculate those errors have also been given in this chapter.

The achievements and contributions of the developed UNLization/IAN module for the Punjabi language have been highlighted in this chapter. This chapter also provide details of UNLization/IAN module of Punjabi language for proposed question answering system.

Chapter 5
UNL Crawler and Optimizer

As discussed in the previous chapters, the question which user asks through the developed QAS is converted to UNL. Before UNL crawler starts finding the answer, this question's UNL and the UNL repository need to be preprocessed. Preprocessing means removing of attributes and scope values from the question's UNL and target UNL repository. After this, UNL crawler is responsible for searching the UNL repository depending on the question's UNL generated in the previous step (UNLization phase). This is called as crawling. This module is a part of the developed web-based QA system interface.

5.1 UNL Crawler (Preprocessing and Crawling Phase)

UNL crawler does two operations *viz.* preprocessing, and crawling for finding the answer. These operations are explained using example sentence (5.1) taken from Agro-Explorer corpus and example sentence (5.4).

ਆਧੁਨਿਕ ਖੇਤੀਬਾੜੀ ਬਹੁਤ ਹੱਦ ਤੱਕ ਇੰਜਿਨਿਯਰਿੰਗ ਅਤੇ ਤਕਨਾਲੋਜੀ ਅਤੇ ਬਾਇਯੋਲਾਜਿਕਲ ਤੇ ਫਿਜ਼ਿਕਲ ਸਾਇੰਸ ਉੱਪਰ ਨਿਰਭਰ ਹੈ

Ādhunika khētībāṛī bahuta hada taka iñjiniyariṅga atē takanālōjī atē bā'iyōlājikala tē phizikala sā'isa upara nirabhara hai

"Modern agriculture depends heavily on engineering and technology and on the biological and physical science". ...(5.1)

UNL of the example sentence (5.1) as given by the UNLization/IAN module (invoked by analysis module) is shown in (5.2) and its UNL graph is depicted in Figure 5.1.

aoj(depend(aoj>thing):0J.@entry.@present.@pred, agriculture(icl>activity):07)
scn(depend(aoj>thing):0J.@entry.@present.@pred, :03)
man(depend(aoj>thing):0J.@entry.@present.@pred, heavily(icl>how):0R)
mod:03(science(icl>art):2S.@entry.@def.@pl, :02)
and:02(physical(aoj>thing):2J.@entry, biological(aoj>thing):24)
and:03(technology(icl>field):1I.@entry, engineering(icl>field):12)
mod(agriculture(icl>activity):07, modern(aoj>thing):00) ...(5.2)

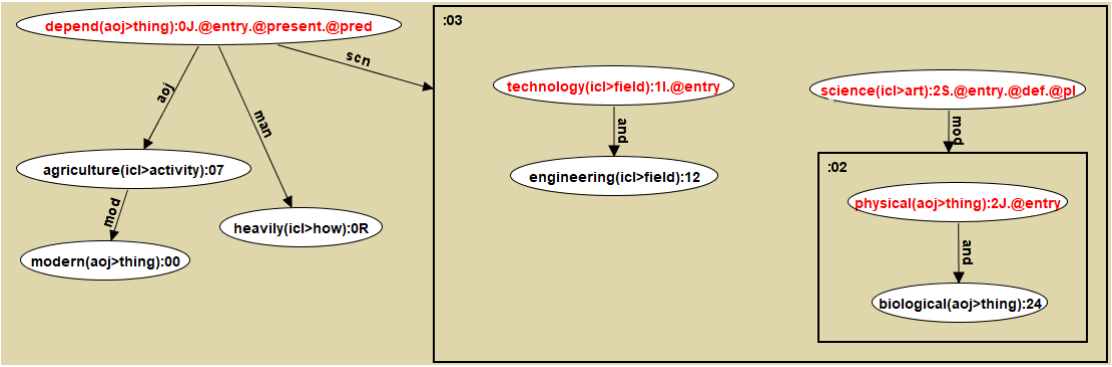


Figure 5.1: UNL Graph of Example Sentence (5.1)

Now, after preprocessing the UNL given in (5.2), its preprocessed UNL is given in (5.3).

{unl}
aoj(depend, agriculture)
scn(depend, :03)
man(depend, heavily)
mod:03(science, :02)
and:02(physical, biological)
and:03(technology, engineering)
mod(agriculture, modern)
{/unl} ...(5.3)

The UNL of this sentence and all other sentences of Agro-Explorer is present in AgroExplorer.txt file which is the part of this complete QAS solution and forms the UNL repository. Let's assume that the user asks the following question.

ਆਧੁਨਿਕ ਖੇਤੀਬਾੜੀ ਕਿਹੜੀ ਚੀਜ਼ ਉੱਪਰ ਨਿਰਭਰ ਹੈ?

Ādhunika khētībāṛī kihaṛī cīza upara nirabhara hai?

"On what does modern agriculture depends?" ...(5.4)

The preprocessed UNL of the input question given in (5.4) is shown in (5.5) and its UNL graph is depicted in Figure 5.2.

{unl}
aoj(depend, agriculture)
scn(depend, 00.@wh)
mod(agriculture, modern)
{/unl} …(5.5)

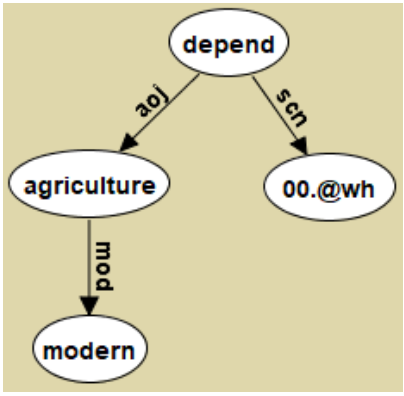


Figure 5.2: UNL Graph of Example Sentence (5.4)

Once preprocessing is completed, UNL crawler parses corpus's UNL and question's UNL and attempts to find the answer. For the current example sentence, the UNL crawler finds answer as "*technology and engineering*", "*science of physical and biological*" as shown in Figure 5.3.

Figure 5.3: Answer Given by UNL Crawler

In the proposed system, the questions asked by the user from the developed question answering system have been categorized into three different types as discussed below.

A. Missing type question

These are the questions in which a node in the UNL graph has an argument "*00.@wh*". For example, consider preprocessed UNL given in (5.6) and its UNL graph shown in Figure 5.4 for the question "*On what does modern agriculture depends*".

{unl}

mod(agriculture, modern)

aoj(depend, agriculture)

scn(depend, 00.@wh)

{/unl} …(5.6)

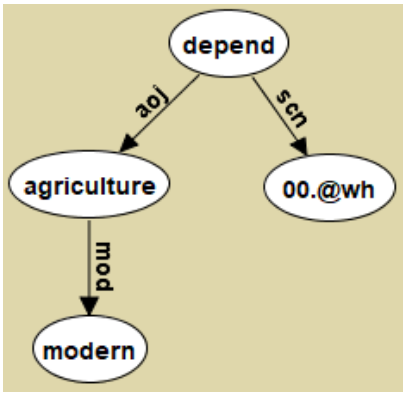


Figure 5.4: UNL Graph of Example Sentence "*On what does modern agriculture depends*"

In such type of questions, UNL crawler works by finding the missing value (as indicated by "*00.@wh*") from the UNL corpus as described in PSEUDOCODE 3 of this chapter.

B. Polar (Yes/No type) question

These are the questions in which a node in UNL graph has an argument "*xyz@YesNo*" and question answering system answers the question as either "*Yes*" or "*No*". Here "*xyz*" means any combination of characters and "*YesNo*" is the attribute which is given during UNLization of the input natural language question sentence. For example, consider preprocessed UNL specified in (5.7) and its UNL graph shown in Figure 5.5 for the question "*Does modern agriculture depends on engineering*".

{unl} mod(agriculture, modern)

aoj(depend, agriculture)

scn(depend, engineering@YesNo)

{/unl} …(5.7)

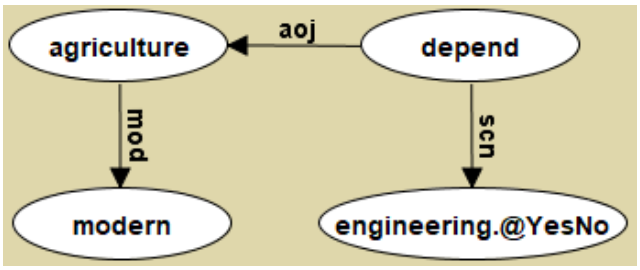


Figure 5.5: UNL Graph of Example Sentence "*Does modern agriculture depends on engineering*"

C. Non-missing type question

These are the questions in which a node in the UNL graph has an argument "*xyz@wh*".

Here, "*xyz*" means any combination of characters (except "*00*" which is used for missing type questions). For example, consider preprocessed UNL given in (5.8) for a question "*What does agriculture include*".

{unl}
aoj(include, agriculture@wh)
{/unl} …(5.8)

In such type of questions, UNL crawler works by creating the answer from the corpus's UNL as described in PSEUDOCODE 3 of this chapter.

Following is the pseudocode used by UNL crawler for crawling.

PSEUDOCODE 1: Crawling

Input: Question's UNL, Corpus's UNL

Output: Answer in English

1. Call the procedure to find question type (PSEUDOCODE 2: FindQuestionType).
2. Call the procedure to get the best possible answer on the basis of type of question (PSEUDOCODE 3: FindAnswer).

The pseudocode to identify the type of question, *i.e.*, if it is a missing type question, Yes/No type question, or non-missing type question is explained as follows.

PSEUDOCODE 2: FindQuestionType

Input: Question's UNL

1. If question's UNL has a node with an argument like '00.@wh' then it's a missing type question.
2. If question's UNL has a node with an argument like 'xyz@YesNo' then it's a polar question.
3. If question's UNL has a node with an argument like 'xyz@wh' then it's a non-missing type question.

The pseudocode for finding the answer is explained as follows.

ALGOR 3: FindAnswer

Input: Question's UNL, Corpus's UNL

Output: Answer in English

1. If the question is a missing type question then
    a. Store name of the relation in which question argument (00.@wh) is present in a variable as answer relation name.
    b. Store the sibling argument of the node in which question argument (00.@wh) is present in a variable as an argument to match.
    c. If 00.@wh is the second argument then do the following:
        i. Crawl the corpus's UNL and look for a node whose relation name is same as that of answer relation name and whose first argument is same as an argument to match.
        ii. Case 1: If such a node is found and the second argument is not a scope then its second argument is the required answer and exit.

            Case 2: Else if such node is found and the second argument is a scope, then resolve that scope using 'hyperNodes' object and exit.

            Case 3: Else if any other nodes are found whose first argument is same as an argument to match, then their second arguments could be a probable answers (later decided by Optimizer) so store them and exit.

            Case 4: Else if no such node is found, then save answer as blank or empty (*i.e.*, answer = "") and exit.
    d. If 00.@wh is the first argument then follow step '1-c' with the only difference that instead of the first argument use second argument and vice-versa and exit.
2. If the question is a polar question then
    a. Store name of the relation in which question argument (xyz@YesNo) is present in a variable as answer relation name. Call this node as candidate node.
    b. Store the sibling argument of the node in which question argument (xyz@YesNo) is present in a variable as an argument to match.
    c. Crawl the corpus's UNL and check if candidate node is present or not. If the candidate node is present then the answer is 'Yes' and exit.

d. If xyz@YesNo is the second argument then do the following:
    i. Crawl the corpus's UNL and look for a node whose relation name is same as that of answer relation name and whose first argument is same as an argument to match.
    ii. If such node is found and the second argument is a scope, then resolve that scope using 'hyperNodes' object and check if 'xyz' is in any of the substrings of that resolved hyperNode.
    iii. If 'xyz' is in any of the substrings of that resolved hyperNode then set answer as 'Yes'

        Else set answer as 'No' and exit.
e. If xyz@YesNo is the first argument then follow step '2-d' with the only difference that instead of the first argument use second argument and vice-versa.

3. If the question is a non-missing type question then
    a. Store name of the relation in which question argument (xyz@wh) is present in a variable as answer relation name.
    b. Store the sibling argument of the node in which question argument (xyz@wh) is present in a variable as an argument to match.
    c. If xyz@wh is the second argument then do the following:
        i. Crawl the corpus's UNL and look for all nodes whose first argument is same as an argument to match, and the second argument is not 'xyz'.
        ii. For all such nodes found if the second argument is not a scope then save their second arguments as the expected required answers and exit.

            Else if the second argument is a scope, then resolve that scope using 'hyperNodes' object and exit.

            Else if no such nodes are found, then save answer as blank or empty (*i.e.*, answer = "") and exit.
    d. If xyz@YesNo is the first argument then follow step '3-c' with the only difference that instead of the first argument use second argument and vice-versa.

Once UNL crawler gives the answer, optimizer analyzes this answer and assigns rank

to it. The working of optimizer has been explained in the following section.

5.2. Optimizer (Optimizing and Ranking Phase)

Optimizer in its optimizing phase analyses the question and tells if the question asked by the user is correct and if its answer can be found. Optimizer also notifies if the answer provided by UNL crawler is a full match/partial match.

As shown in Figure 5.1, it's the optimizer which tells that "*it's a perfect match. Your answer is*". After optimizing phase, in the ranking phase, optimizer gives the following rank to the answer found by the UNL crawler:

- **Rank 1**

  If all the information in question's UNL is present in corpus's UNL and answer given by UNL crawler is not blank, then optimizer assigns rank 1 to the answer and says that "*it's a perfect match. Your answer is:*".

- **Rank 2**

  If not all the information in question's UNL is present in corpus's UNL and answer is not blank, then optimizer assigns rank 2 to the answer and reports that "*It's ALMOST a perfect match. We cannot find some information in Database*".

- **Rank 3**

  If there is a mismatch between the information in question's UNL and information present in corpus's UNL and answer given by UNL crawler is not blank, then optimizer assigns rank 3 to the answer and reports that "*It's a partial match. The most probable answer is*".

- **Rank 4**

  If all the information in question's UNL is present in corpus's UNL but the answer given by UNL crawler is blank, then optimizer assigns rank 4 to the answer and reports that "*Your question is correct but we are sorry because our database does not contain sufficient information to answer this*".

- **Rank 5**

  If answer given by UNL crawler is blank, then optimizer assigns rank 5 to the answer and reports "*Cannot find the answer*".

Pseudocode for optimization and ranking is explained as follows.

PSEUDOCODE 4: Optimization_Ranking

Input: Question's UNL, Corpus's UNL, List of candidate answers given by UNL crawler

Output: Answers with ranking and relevance in the proper format

1. If the question's UNL does not contain any relation having node '00.@wh' (indicating missing type question) or 'xyz@wh' (indicating non-missing type question), or a node having attribute '@YesNo' (indicating polar type question), then Optimizer reports '*Sorry we cannot find any question'*.
2. If the question is a missing type question then
   a. Parse the question's UNL and corpus's UNL.
   b. If all the relations and their arguments in question's UNL are also present in corpus's UNL, then Optimizer assigns rank 1 to the answer given by UNL crawler and reports that '*it's a perfect match. Your answer is:*'.
   c. If:
      i. There are some relations which are present in question's UNL but not present in corpus's UNL,
      AND
      ii. All the other relations and their arguments in question's UNL are present in corpus's UNL,
      Then Optimizer assigns rank 2 to the answer given by UNL crawler and reports that '*It's ALMOST a perfect match. We cannot find some information in Database*'.
   d. If:
      i. All the relations in question's UNL are also present in corpus's UNL,
      AND
      ii. There exist few relations in question's UNL whose arguments are not matching with the corpus's UNL,
      Then Optimizer assigns rank 3 to the answer given by UNL crawler and reports that '*It's a partial match. The most probable answer is*'.

e. If:

   i. All the information in question's UNL is present in corpus's UNL,

      AND

   ii. The UNL crawler cannot find the answer and gives answer as blank,

      Then Optimizer assigns rank 4 to the answer and reports that '*Your question is correct but we are sorry because our database does not contain sufficient information to answer this*'.

Else

Optimizer assigns rank 5 to the answer and reports '*Cannot find the answer*'.

Consider an example sentence given in (5.9) taken from Agro-Explorer.

Plant breeding and genetics contribute immeasurably to farm productivity during summer. ...(5.9)

UNL of this example sentence (5.9) is specified in (5.10) and its UNL graph is shown in Figure 5.6.

{unl}
aoj(contribute,:01)
gol(contribute,productivity)
man(contribute,immeasurably)
mod(productivity,farm)
and:01(genetics,breeding)
mod:01(breeding,plant)
tim(contribute,summer)
{/unl} ...(5.10)

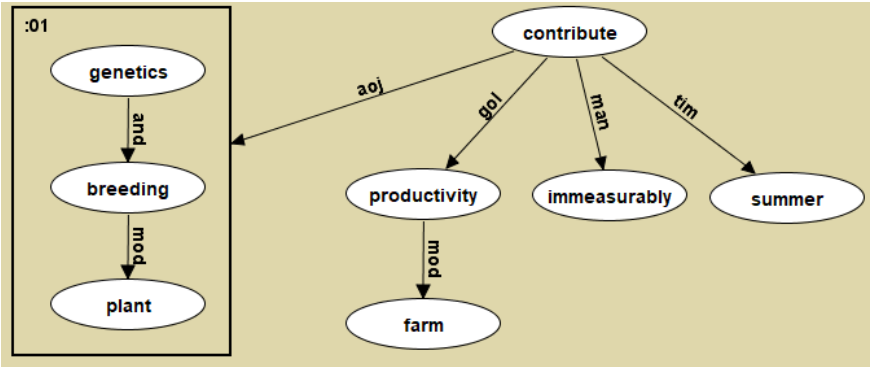


Figure 5.6: UNL Graph of Example Sentence (5.9)

For the given example sentence (5.9), consider the following cases which explain the working of optimizer for all the steps given in PSEUDOCODE 4.

Case 1: Let's say the user asks the question given in (5.11).

What contributes immeasurably to farm productivity during summer ? …(5.11)

UNL of this question is specified in (5.12) and its UNL graph is shown in Figure 5.7.

{unl}

aoj(contribute,00.@wh)

gol(contribute,productivity)

man(contribute,immeasurably)

mod(productivity,farm)

tim(contribute,summer)

{/unl} …(5.12)

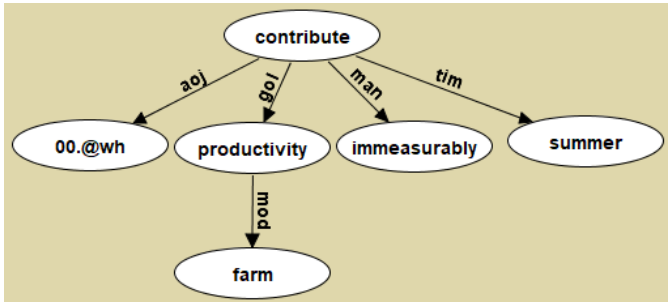


Figure 5.7: UNL Graph of Example Sentence (5.11)

UNL crawler gives the output as follows.

1. *genetics and breeding*
2. *breeding of plant*

Optimizer analyses this output given by UNL crawler and finally displays the result as follows.

Its a perfect match. Your answer is:

1. *genetics and breeding*
2. *breeding of plant*

Optimizer gives rank 1 to this answer because all the information asked in the question (5.11) is present in the corpus (UNL given in (5.10) in this case).

Case 2: Let's say the user asks the question given in (5.13).

What contributes immeasurably to farm productivity in India ? …(5.13)

UNL of this question is specified in (5.14) and its UNL graph is shown in Figure 5.8.

{unl}
aoj(contribute,00.@wh)
gol(contribute,productivity)
man(contribute,immeasurably)
mod (productivity,farm)
plc (productivity,India)
{/unl} …(5.14)

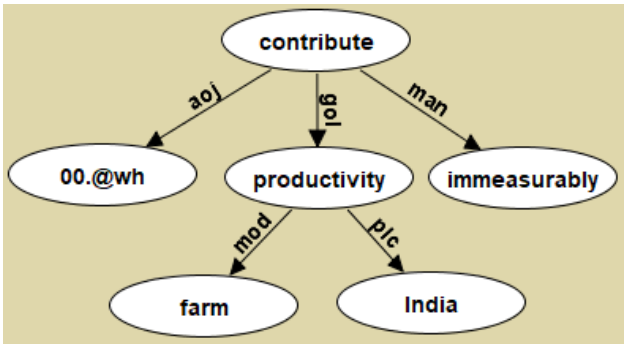


Figure 5.8: UNL Graph of Example Sentence (5.13)

UNL crawler gives the output as follows.

1. *genetics and breeding*
2. *breeding of plant*

Optimizer analyses this output given by UNL crawler and finally displays the result as follows.

It's ALMOST a perfect match. We cannot find some information in the Database. The answer is:

1. *genetics and breeding*
2. *breeding of plant*

Optimizer gives rank 2 to this answer because all the information asked in the question (place of productivity, *i.e.*, India) is not present in the corpus (UNL given in (5.10) in this case).

Case 3: Let's say the user asks the question given in (5.15).

What contributes immeasurably to farm productivity during winter ? …(5.15)

UNL of this question is specified in (5.16) and its UNL graph is shown in Figure 5.9.

{unl}
aoj(contribute,00.@wh)
gol(contribute,productivity)
man(contribute,immeasurably)
mod(productivity,farm)
tim(contribute,winter)
{/unl} …(5.16)

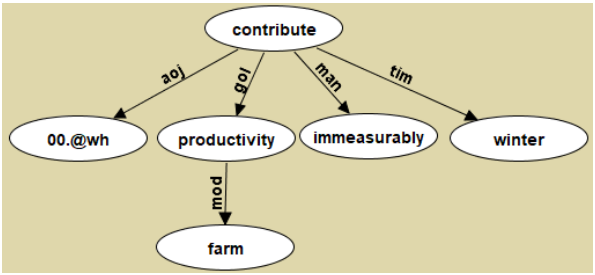


Figure 5.9: UNL Graph of Example Sentence (5.15)

UNL crawler gives the output as follows.

1. *genetics and breeding*
2. *breeding of plant*

Optimizer analyses this output given by UNL crawler and finally displays the result as follows.

Its a partial match. The most probable answer is:

1. *genetics and breeding*
2. *breeding of plant*

Optimizer gives rank 3 to this answer because of the difference between "*summer*" and "*winter*" in corpus's UNL and question's UNL respectively.

Case 4: Reconsider the example given in (5.11) and assume that the user asks the same question as given in (5.17).

What contributes immeasurably to farm productivity during summer ? …(5.17)

UNL of this question is specified in (5.18) and its UNL graph is shown in Figure 5.10.

{unl}

aoj(contribute,00.@wh)

gol(contribute,productivity)

man(contribute,immeasurably)

mod(productivity,farm)

{/unl} …(5.18)

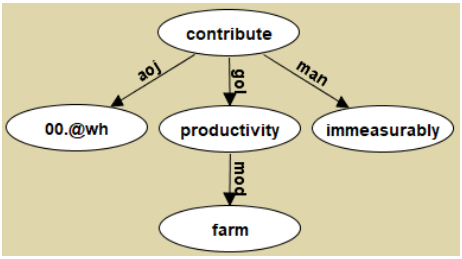


Figure 5.10: UNL Graph of Example Sentence (5.17)

Additionally, let us assume that the UNL of example (5.9) is specified in (5.19) and its UNL graph is shown in Figure 5.11 does not contain the striked node.

{unl}

~~aoj(contribute,:01)~~

gol(contribute,productivity)

man(contribute,immeasurably)

mod(productivity,farm)

and:01(genetics,breeding)

mod:01(breeding,plant)

tim(contribute,summer)

{/unl} ...(5.19)

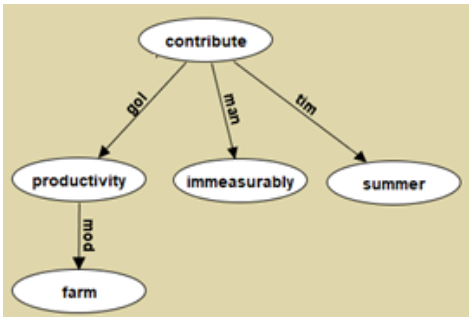


Figure 5.11: UNL Graph of UNL Given in (5.19)

In this case, the UNL crawler will give the output as blank. Optimizer will analyse this output, gives rank 4 to it and finally displays the result as "*Your question is correct but we are sorry because our database does not contain sufficient information to answer this*".

Case 5: Let's say the user asks the question given in (5.20).

How cotton is produced? ...(5.20)

UNL of this sentence (5.20) is specified in (5.21).

gol(produce,cotton);
aoj(produce,00.@wh); …(5.21)

In this case, the UNL crawler will give the output as blank. Optimizer will analyse this output, gives rank 5 to it and finally displays the result as "*Cannot find the answer*".

If the user has not asked any question (identified by Optimizer through question's UNL), then optimizer makes sure that no further processing is done and it stops the execution and searching/finding process immediately by alerting the user that "*Sorry we cannot find any question*".

Let's say the user asks the question given in (5.22).

Plant breeding and genetics contribute immeasurably to farm productivity during summer. ...(5.22)

UNL of this sentence (5.22) is specified in (5.23) and its UNL graph is shown in Figure 5.12.

{unl}
aoj(contribute,:01)
gol(contribute,productivity)
man(contribute,immeasurably)
mod(productivity,farm)
and:01(genetics,breeding)
mod:01(breeding,plant)
tim(contribute,summer)
{/unl} ...(5.23)

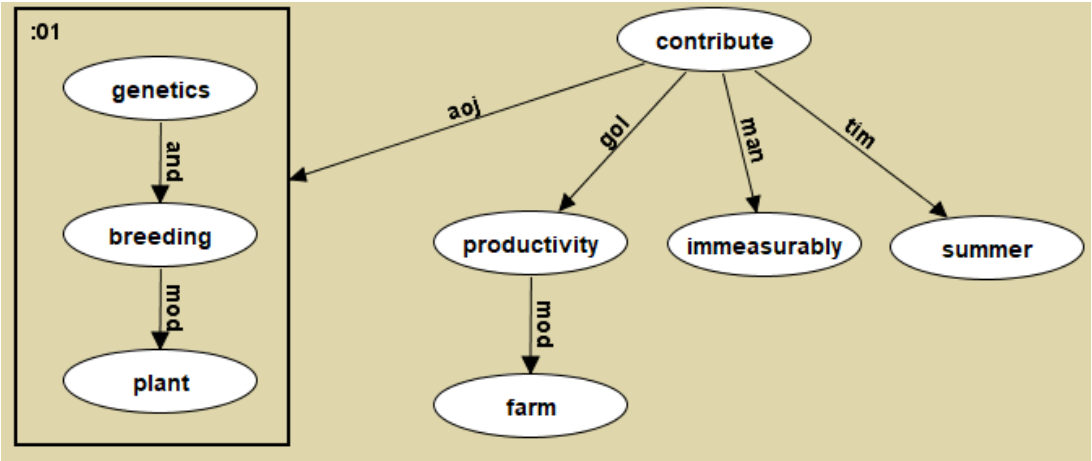


Figure 5.12: UNL Graph of Example (5.22)

Now, since this question's UNL does not contain a question node (*i.e.*, a node with attribute '00.@wh'), therefore the optimizer makes sure that no further processing is done and it stops the execution and searching/ finding process immediately by alerting the user that "*Sorry we cannot find any question*".

After optimizer assigns rank to the answer given by UNL crawler, this answer is converted to UNL and sent to generation module which invokes NLization/EUGENE module to get the final answer in target natural language. The NLization/EUGENE module has been discussed in the next chapter.

Chapter Summary

In this chapter, the detailed information of UNL crawler and optimizer has been documented. Initially, in this chapter the concept of preprocessing has been explained

using example sentences. This chapter also documents the crawling, optimizing, and ranking processes. The pseudocodes of crawling, optimizing, and ranking have been explained in this chapter.

In the proposed system, the questions asked by the user from the developed question answering system have been categorized into three different types, *i.e.*, missing type question, polar (Yes/ No type) question, and non-missing type question. This chapter explains these question types with the help of example sentences.

After optimizing phase, in the ranking phase, optimizer gives rank 1, rank 2, rank 3, rank 4, or rank 5 to the answer found by the UNL crawler. In this chapter, the ranking mechanism used by the optimizer has been explained for every case with the help of example sentences. If all the information in question's UNL is present in corpus's UNL and answer given by UNL crawler is not blank, then optimizer says that "*it's a perfect match. Your answer is:*". If not all the information in question's UNL is present in corpus's UNL and answer is not blank, then optimizer reports that "*It's ALMOST a perfect match. We cannot find some information in Database*". If there is a mismatch between the information in question's UNL and information present in corpus's UNL and answer given by UNL crawler is not blank, then optimizer reports that "*It's a partial match. The most probable answer is*". If all the information in question's UNL is present in corpus's UNL but the answer given by UNL crawler is blank, then optimizer reports that "*Your question is correct but we are sorry because our database does not contain sufficient information to answer this*". If answer given by UNL crawler is blank, then optimizer reports "*Cannot find the answer*".
Optimizer also makes sure that if user didn't ask any question, then no further processing is done and it stops the execution and searching/finding process immediately by alerting the user that "*Sorry we cannot find any question*".

Chapter 6

EUGENE/NLization Module for NLization of Punjabi Language

This chapter focuses on the NLization process and generation module in depth. NLization is done with the help of online tool EUGENE (*i.e.*, dEep-to-sUrface GENErator) developed by UNDL foundation to convert UNL to NL text. The way analysis module of the developed question answering system invokes IAN, its generation module invokes EUGENE/NLization module to convert the answer into the target language (Punjabi in this case).

6.1 EUGENE Framework

EUGENE is a natural language generator which is fully automatic. It takes UNL as an input and gives an output in the form of natural language. It is language-independent. It uses the UNL-NL dictionary and a grammar to process the UNL which are provided as separate files. The NLization process, *i.e.*, converting input UNL to target natural language, is shown in Figure 6.1.

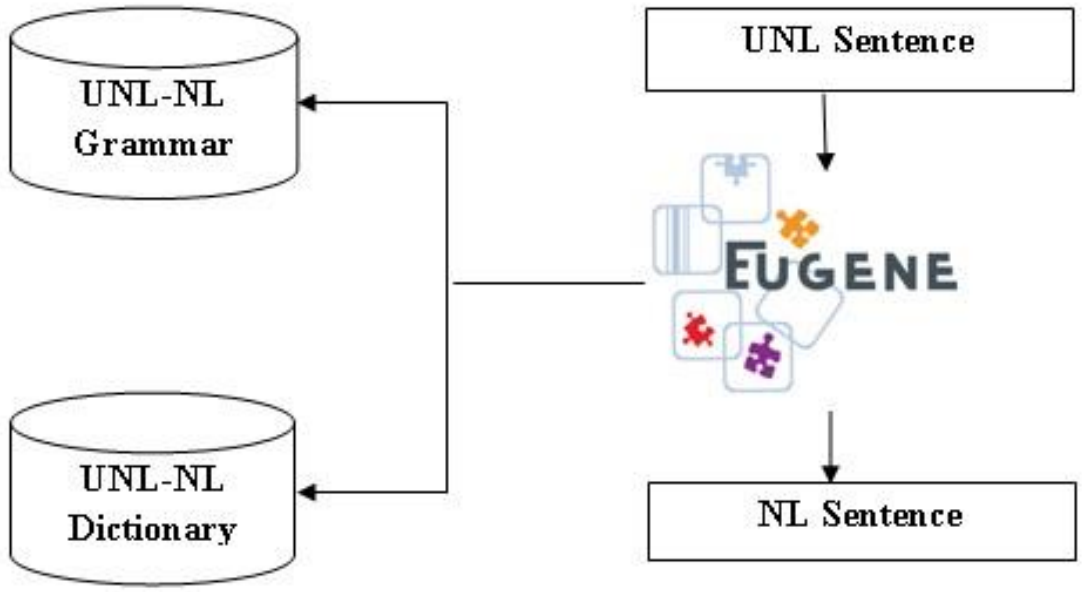


Figure 6.1: Processing of UNL Sentence

EUGENE performs the three operations over the input files which are briefly discussed as follows.

- Segmentation - In this process, the input document is divided into a number of isolated graphs and each graph is processed one at a time.

- Tokenization - In this process, tokens of each graph are identified. Tokens here refer to UWs, relations, and attributes.
- Transformation – After tokenization, transformation rules of the grammar are applied over each graph to generate a NL sentence.

The fully automated web-based EUGENE tool developed by UNDL foundation provides six tabs to accomplish the desired task of NLization, *i.e.*, UNL input, Dictionaries tab, T-Rules (Transformation Rules) tab, D-Rules (Disambiguation Rules) tab, EUGENE console, and Compare tab. The functions and snapshots of these tabs are explained as follows.

The first tab is the UNL Input tab. In this tab, a user can view UNL documents, load UNL documents, edit UNL expressions, delete existing documents, and can share files with multiple users. The snapshot of UNL input tab is shown in Figure 6.2.

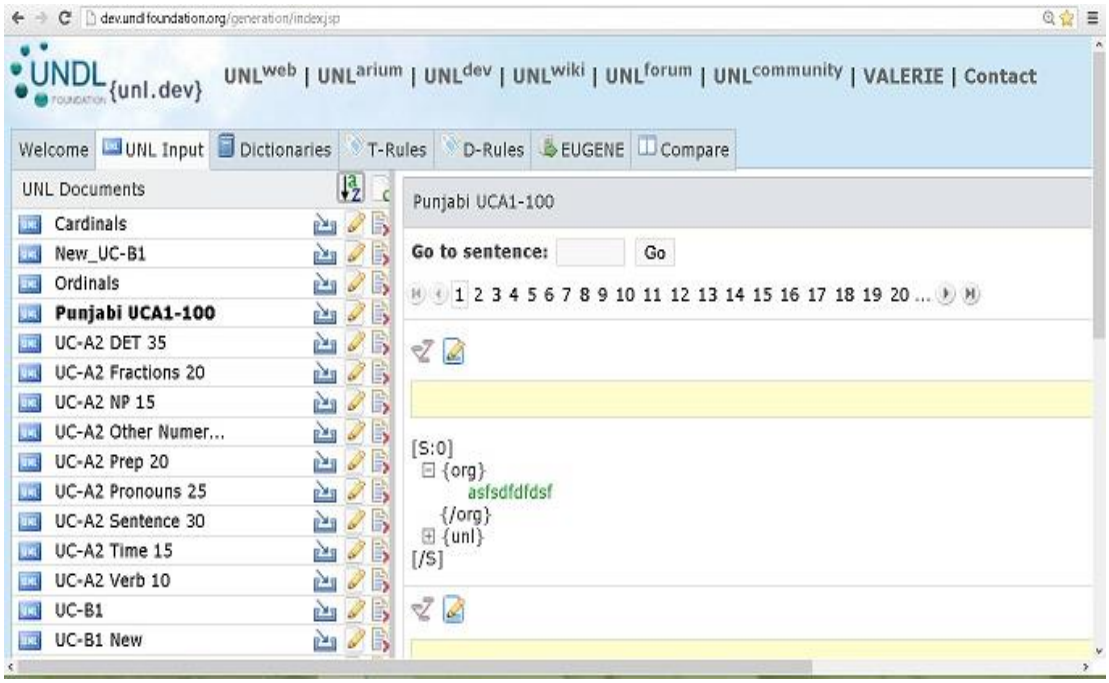


Figure 6.2: Snapshot of UNL Input Tab in EUGENE [47]

The second tab is the Dictionary tab. In this tab, UNL-NL dictionaries which are used in natural language generation have been created. A user can either create a new file or upload an existing file of the dictionary. Once a dictionary is created/uploaded, it is ready for the use. Any number of different dictionaries can be loaded to process the same corpus. The snapshot of the dictionary tab is shown in Figure 6.3.

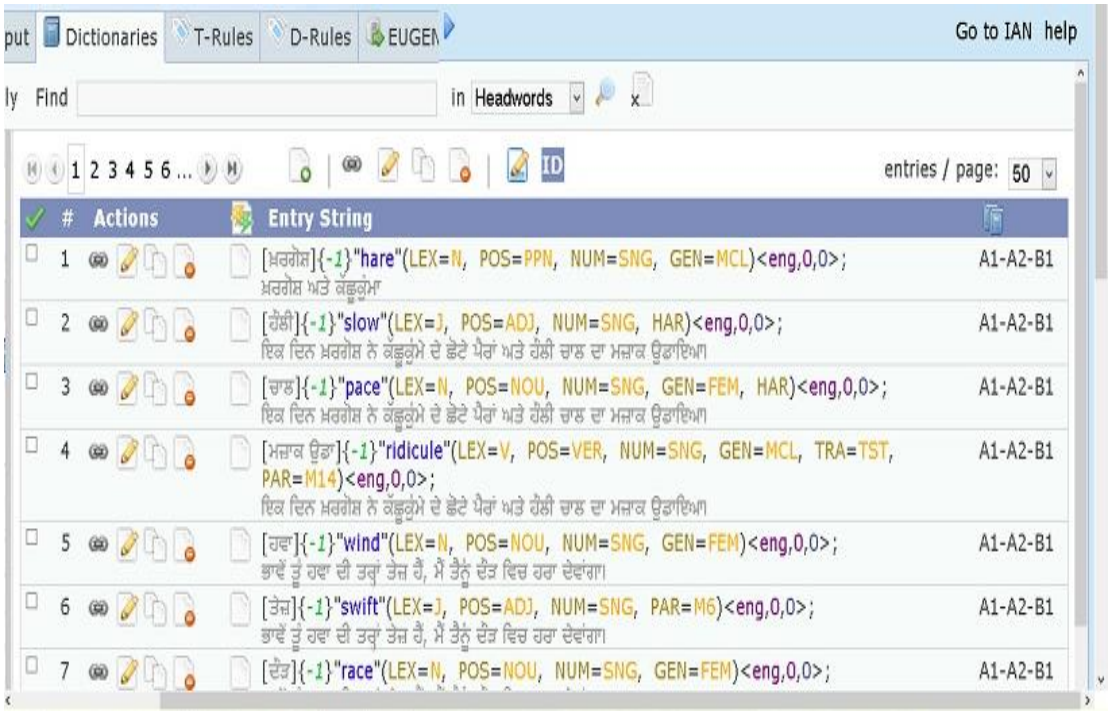


Figure 6.3: Snapshot of Dictionary Tab in EUGENE [47]

The third tab is T-Rules or TRules tab. In this tab, UNL-NL transformation grammar rules which are used to convert the UNL input into the NL output have been created. A user can either create a new file or upload an existing file of transformation grammar rules. In this tab, the transformation grammar must agree with the UNL grammar specifications and must consist of only transformation rules. Any number of different grammars can be loaded to process the same corpus. The snapshot of T-Rules tab is shown in Figure 6.4.

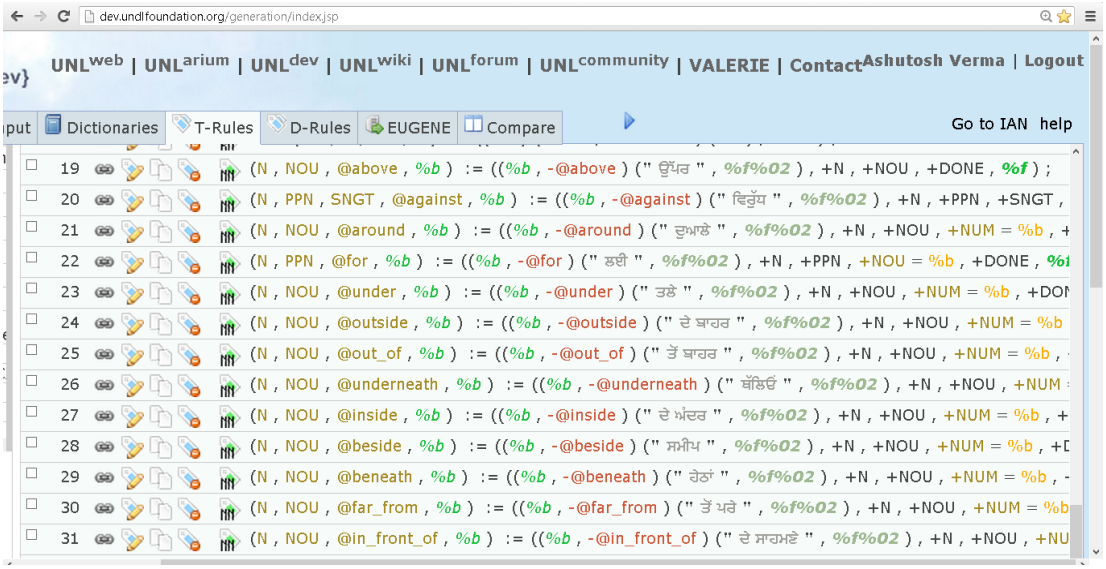


Figure 6.4: Snapshot of Transformation Rules in EUGENE [47]

The fourth tab is D-Rules or DRules tab. In this tab, UNL-NL disambiguation grammar is created. This grammar is mainly used to control the tokenization as well as to improve the output of transformation grammar. Any number of different grammars can be loaded to process the same corpus. The snapshot of T-Rules tab is shown in Figure 6.5.

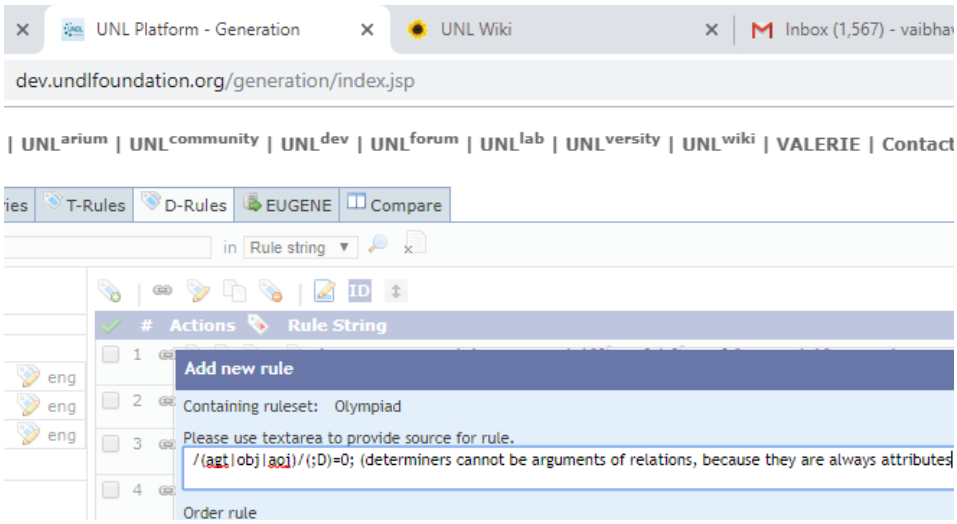


Figure 6.5: Snapshot of DRules Tab [47]

The fifth tab is EUGENE console tab. This tab displays the final output in the form of natural language generated sentence. It brings the list of sentences. These sentences can be processed one at a time or in a range. The final output is shown in five different trace levels. The snapshot of EUGENE console is shown in Figure 6.6.

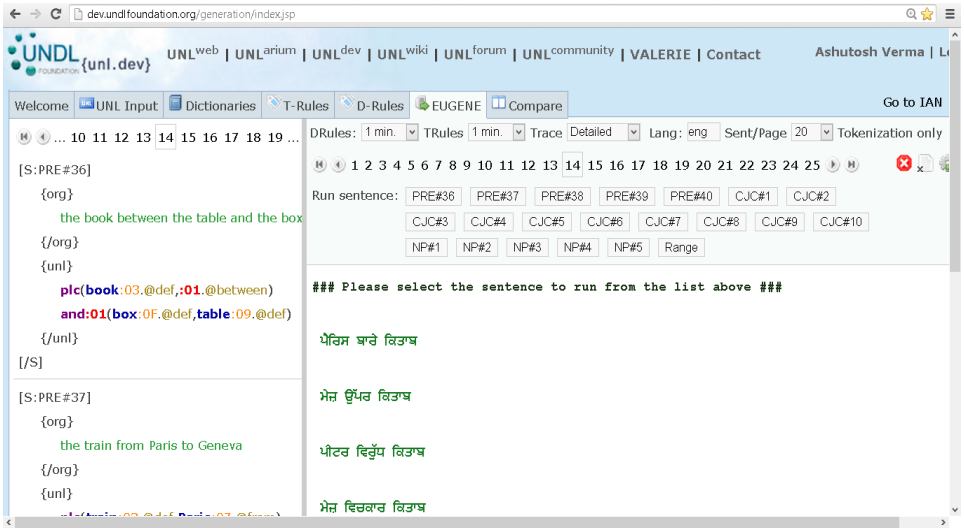


Figure 6.6: Snapshot of Getting NL Sentence by Processing UNL Sentence one at a Time [47]

The last tab is the compare tab which has the ability to compare the final two results.

6.2 Working of EUGENE

The working of EUGENE is explained with the help of seven example sentences for each of the following part of speeches.

- Adjectives
- Conjunctions
- Determiners
- Verbs
- Nouns
- Pronouns
- Prepositions

The TRules used for NLization by EUGENE involve the use of generation dictionary and inflectional paradigms. Unlike analysis dictionary, generation dictionary contains the entry of only root word. The inflected forms of that root word are obtained with the help of inflection paradigms.

Consider, for instance, the case of the English nouns making the plural in –s (*book>books, table>tables, etc.*). This can be expressed by affixation (suffixation of -s) and it is regular (there are many words in this set). This morphological behavior is then described by the paradigm M2, which contains two rules as follows.

- SNG:=0>""; (do not add anything to the word in case of singular)
- PLR:=0>"s"; (add "s" to the end of the word in case of plural)

This paradigm is created within the natural language grammar and is associated, in the natural language dictionary, to all words having the same morphological behavior.

The TRule for paradigm M2 would be:

- (%x,M2):= (%x, - M2, FLX(SNG:=0>""; PLR:=0>"s"));

It means that if a node is having paradigm M2, then remove that paradigm (as indicated by "*–M2*") so as to remove recursion, and if this node is having "*SNG*" (*i.e.,* Singular) attribute then do nothing but if this node is having "*PLR*" (*i.e.,* Plural) attribute then add "*s*" in the root word by not removing any character (as indicated by "*0*") from the last. The "*FLX*" indicates that word will be processed for inflection as per this given rule. In Table 6.1 (for example sentence 6.1) firing of a paradigm has been explained.

In the generation dictionary the root word enteries are as follows.

- [table]{ID}"UW"(LEX=N,POS=NOU,...,PAR=M2)<eng,0,0>;
- [book]{ID}"UW"(LEX=N,POS=NOU,...,PAR=M2)<eng,0,0>;

The entire NLization process involving the use of generation dictionary, TRules, DRules, and inflection paradigms has been explained for Punjabi language with the help of example sentences for each of the above mentioned part of speeches in the follwoing subsections.

6.2.1 NLization of Adjectives

The NLization of adjectives in UNL sentence to NL sentence is presented using example given in (6.1).

English sentence: “The beautiful cars” …(6.1)

UNL of (6.1) is given in (6.2).

{unl}
mod(car:03.@def.@pl, beautiful:01.@pl)
{/unl} …(6.2)

Expected Punjabi sentence: ਸੋਹਣੀਆਂ ਗੱਡੀਆਂ …(6.3)

Transliterated Punjabi sentence: *sohnian gaddian*

All nodes (or UWs) identified in UNL sentence given in (6.2), are mapped with their corresponding dictionary entries and are listed below.

beautiful:01[ਸੋਹਣੀ]{}"beautiful"(LEX=J, POS=ADJ, GEN=FEM, NUM=SNG, PAR= M3);
car:03 - [ਗੱਡੀ]{}"car" (LEX=N,POS=NOU,GEN=FEM,NUM=SNG,PAR=M3);

Here, “*LEX”* represents lexical category, “*J*” represents adjective, “*POS*” represents part-of-speech, “*ADJ*” represents adjective, “*GEN*” represents gender which can be either “*MCL*” for male and “*FEM*” for female, “*NUM*” represents number whose value could be either “*SNG*” for singular or “*PLR*” for plural, “*PAR*” represents the paradigm number which is “*M3*” in above cases, it specifies which inflection paradigm has to be fired for inflecting the root words. “*N”* represents a noun. The process of NLization of example UNL sentence given in (6.2) has been illustrated in Table 6.1. Here, transliteration of each Punjabi word is also mentioned.

Table 6.1: NLization Process of Example Sentence (6.2)

Rule Fired	Description	Action Taken
(%x,@def,@pl):=(%x,-@def, -@pl, -NUM, +NUM=PLR)	This rule will remove the attributes "@def" and "@pl" and will result into the same node with "PLR" as an attribute.	To: [car:03.@def.@pl] Result: ["car:03"] *["gaddi:03]*
(%x,@pl):=(%x, -@pl, -NUM, +NUM=PLR)	This rule will remove the attribute "@pl" and will result into the same node with "PLR" as an attribute.	To: [beautiful:01.@pl] Result: ["beautiful:01"] *["sohni:01]*
(%y,M3):=(%y,+FLX(SNG:=0>"";PLR:=0>"ਆਂ";MCL:=1>"ਆਂ"),- M3);	This paradigm M3 attaches the corresponding postfix to verbs in Punjabi. The attribute "FLX" indicates that node will be handled for inflection according to the given rule.	To: [car:03] Result: ["ਗੱਡੀਆਂ ":03] *["gaddian":03]* "ਆਂ" "*an*" is appended to UW "ਗੱਡੀ" "*gaddi*", because of paradigm M3 and "PLR" attribute.
(%y,M3):=(%y,+FLX(SNG:=0>"";PLR:=0>"ਆਂ";MCL:=1>"ਆਂ"),- M3);	This paradigm M3 attaches the corresponding postfix to verbs in Punjabi. The attribute "FLX" indicates that node will be handled for inflection	To: [beautiful:01] Result: ["ਸੋਹਣੀਆਂ ":01] *["sohnian":01]* "ਆਂ" "*an*" is appended to UW "ਸੋਹਣੀ" "*sohni*",

	according to the given rule.	because of paradigm M3 and “PLR” attribute.
mod(%a,N;%b,J):= (%b)(" ")(%a);	This T-rule has been defined to resolve the “mod” relation between noun and adjective.	To: mod(car:03, beautiful:01) Result: [sc:01(#L:01(-:02, beautiful:01); #L:01(beautiful:01, -:04); #L:01(-:04, car:03))] Intermediate Result: ਸੋਹਣੀ ਗੱਡੀ *sohni gaddi* As a result, this relation is resolved by placing adjective before noun with space between them.
({N|V|D|J|R},FLX,^inflected, %x):=(!FLX,-FLX,+inflected ,%x);	This rule fires the corresponding paradigm (as directed by “*!FLX*”) to inflect the root word “ਸੋਹਣੀ” “*sohni”* based on number and gender information associated with the word.	To: [beautiful:01] Result: ["**ਸੋਹਣੀਆਂ "**:01] *["sohnian"**:01]*** **"ਆਂ" "***an”* is appended to UW “ਸੋਹਣੀ” “*sohni”*, because this root word has “PLR” attribute.
({D|N|R|V|J}, ^inflected, FLX,%y):=(!FLX, +inflected, -FLX,%y);	It fires the corresponding paradigm to inflect the root word “ਗੱਡੀ” “*gaddi”* based on number and gender	To: [car:03] Result: [**"ਗੱਡੀ":**03] *["gaddi**":03]*** Final Output: ਸੋਹਣੀਆਂ ਗੱਡੀਆਂ

	information associated with the word.	*sohnian gaddian* "ਆਂ" "*an*" is appended to UW "ਗੱਡੀ" "*gaddi"*, because this root word has "PLR" attribute.

The natural language output generated by EUGENE for sentence (6.1) is given in (6.4) and it corresponds to the UNL sentence given in (6.2).

{org}
The beautiful cars
{/org}
{pun} ਸੋਹਣੀਆਂ ਗੱਡੀਆਂ {/pun}

{unl}
[W]mod(car:03.@def.@pl, beautiful:01.@pl)[/W]
{/unl} …(6.4)

6.2.2 NLization of Conjunctions

The NLization of conjuctions in UNL sentence to NL sentence is presented using example given in (6.5).

English sentence: *"*John, Mary, Peter and Paul*"* …(6.5)

UNL of (6.5) is given in (6.6) and its UNL graph is shown in Figure 6.7.

{unl}
and(Paul, :02)
and:02(Peter, :01)
and:01(Mary, John)
{/unl} …(6.6)

Expected Punjabi sentence: ਜਾਨ, ਮੈਰੀ, ਪੀਟਰ ਅਤੇ ਪਾਲ …(6.7)

Transliterated Punjabi sentence: *jan, mari, peetar ate pal*

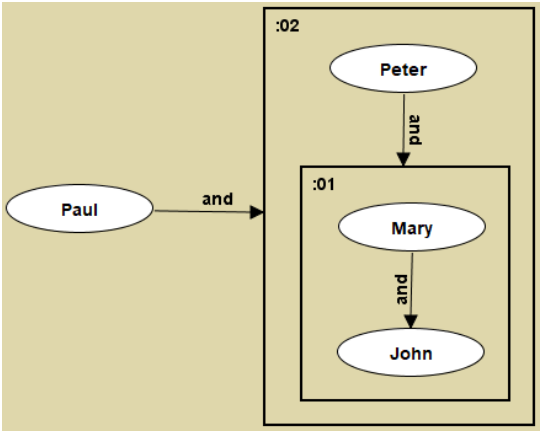


Figure 6.7: UNL Graph of Example (6.5)

In UNL sentence (6.6), last "*and*" relation has two nouns, which are "*Mary*" and "*John*". This "*and*" relation is a scope node ":01" and acts as the object in another "*and*" relation, which itself is a scope node ":02" and acts as the object in another "*and*" relation to form a complete and meaningful sentence.

All nodes (or UWs) identified in UNL sentence given in (6.6), are mapped with their corresponding dictionary entries and are listed below.

Mary:01 - [ਮੈਰੀ]{}"Mary"(LEX=N,POS=NOU,GEN=FEM,NUM=SNG);

John:03 - [ਜਾਨ]{}"John"(LEX=N,POS=NOU,GEN=MCL,NUM=SNG);

Peter:02 - [ਪੀਟਰ]{}"Peter"(LEX=N,POS=NOU,GEN=MCL,NUM=SNG);

Paul:04 - [ਪਾਲ]{}"Paul"(LEX=N,POS=NOU,GEN=MCL,NUM=SNG);

The process of NLization of example UNL sentence given in (6.6) has been illustrated in Table 6.2. The transliteration of each Punjabi word is also mentioned in the table.

Table 6.2: NLization Process of Example Sentence (6.6)

Rule Fired	Description	Action Taken
and(%x;%y):=((%y, +>BLK)([and],LEX=C, POS=CCJ,+>BLK)(%x, +>BLK),+LEX=%x);	This T-rule has been defined to resolve the "and" relation between two UWs. It adds ">BLK" attribute with each word for inserting blank spaces between the words	To: and(Paul:04, :02) Result: [sc:03(#L:03(:02, and:06); #L:03(and:06, Paul:04))] Intermediate Result: :02and ਪਾਲ *:02and pal*

	later on. Also adds "and" between the words which can be translated to Punjabi, later on with direct mapping.	As a result, first relation "and" is resolved; scope node ":02" to be resolved is placed before "Paul" with word "and" as conjunction between them.
and(%x;%y):=((%y, +>BLK)([and],LEX=C, POS=CCJ,+>BLK)(%x, +>BLK),+LEX=%x);	This T-rule has been defined to resolve the "and" relation between two UWs.	To: and:02(Peter:02, :01) Result: [sc:04(#L:04(:01, and:08); #L:04(and:08,Peter:02))]y); Intermediate Result: :01andਪੀਟਰandਪਾਲ *:01andpeetarandpal* As a result, second relation "and" is resolved; scope node ":01" to be resolved is placed before "Peter" with word "and" as conjunction between them.
and(%x;%y):=((%y, +>BLK)([and],LEX=C, POS=CCJ,+>BLK)(%x, +>BLK),+LEX=%x);	This T-rule has been defined to resolve the "and" relation between two UWs.	To: and:01(Mary:01, John:03) Result: [sc:05(#L:05(John:03,and :0A); #L:05(and:0A, Mary:01))] Intermediate Result: ਜਾਨandਮੈਰੀandਪੀਟਰandਪਾਲ *janandmariandpeetarandpal* As a result, relation "and" is resolved; "John" is placed before "Mary" with word "and" as conjunction between them.

([and],%c1)(%x,^ADJ) ({[and]|COMMA},%c2):=([,],",",+PUT=COMMA, +> BLK,%c3)(%x)(%c2);	This T-rule has been defined to add “, ” (comma and a blank space) before a node which does not have “ADJ” attribute and its subsequent node is either “and” or “comma”. This rule will keep on firing itself till it finds three nodes in this form “([and],%c1)(%x) ({[and]|COMMA},%c2)” for transforming “and” to “, ”. So, this rule will be fired for all pairs of UWs having “and” as conjunction between them, except for last pair.	To: #L(and:0A, Mary:01); #L(Mary:01, ,:0B) Result: #L(, :0C, Mary:01); #L(Mary:01, , :0B) Intermediate Result: ਜਾਨ, ਮੈਰੀandਪੀਟਰandਪਾਲ *jan, mariandpeetarandpal* As a result, “, ” (comma and a blank space) is inserted between “John” and “Mary”.
([and],%c1)(%x,^ADJ) ({[and]|COMMA},%c2):=([,],",",+PUT=COMMA, +>BLK,%c3)(%x)(%c2);	This T-rule has been defined to add “, ” (comma and a blank space) before a node which does not have “ADJ” attribute and its subsequent node is either “and” or “comma”.	To: #L(and:09, Peter:02); #L(Peter:01, ,:0D) Result: #L(, :0E, Peter:02); #L(Peter:02, , :0D) Intermediate Result: ਜਾਨ, ਮੈਰੀ, ਪੀਟਰandਪਾਲ *jan, mariandpeetarandpal*

		As a result, “, ” (comma and a blank space) is inserted between “Mary” and “Peter”.
(%x,>BLK)(%y,^BLK, ^PUT,^STAIL):=(%x,->BLK)(" ",+BLK)(%y);	This T-rule has been defined to add blank space after a node having “>BLK” attribute and its subsequent node does not have “BLK”, “PUT” and “STAIL” attributes. Also, “>BLK” attribute is removed from the first node so that this relation should not get executed for this UWs.	To: #L(Peter:02, and:06) Result: #L(Peter:02, -:0P); #L(-:0P, and:06) Intermediate Result: **ਜਾਨ**, **ਮੈਰੀ**, **ਪੀਟਰ** and**ਪਾਲ** *jan, mari, peetar andpal* As a result, a blank space is inserted between “Peter” and “and”.
(%x,>BLK)(%y,^BLK, ^PUT,^STAIL):=(%x,->BLK)(" ",+BLK)(%y);	This T-rule has been defined to add blank space after a node having “>BLK” attribute and its subsequent node does not have “BLK”, “PUT” and “STAIL” attributes.	To: #L(and:06, Paul:04) Result: #L(and:06, -:0Q); #L(-:0Q, Paul:04) Intermediate Result: **ਜਾਨ**, **ਮੈਰੀ**, **ਪੀਟਰ** and **ਪਾਲ** *jan, mari, peetar and pal* As a result, a blank space is inserted between “and” and “Paul”.
([and]):=([**ਅਤੇ**]);	This T-rule generates Punjabi for the word “and” with direct mapping.	To: [and:04] Result: [**"ਅਤੇ":**04] *["**ate":**04]*

		Final Output: ਜਾਨ, ਮੈਰੀ, ਪੀਟਰ ਅਤੇ ਪਾਲ *jan, mari, peetar ate pal* "and" gets translated to "ਅਤੇ" "*ate*" directly.

The natural language output generated by EUGENE for sentence (6.5) is given in (6.8), correspond to UNL sentence given in (6.6).

{org}
John, Mary, Peter and Paul
{/org}
{pun}
ਜਾਨ, ਮੈਰੀ, ਪੀਟਰ ਅਤੇ ਪਾਲ
{/pun}
{unl}
[W]
and(Paul, :02)
and:02(Peter, :01)
and:01(Mary, John)
[/W]
{/unl} …(6.8)

6.2.3 NLization of Determiners

The NLization of determiners in UNL sentence to NL sentence is presented using an example given in (6.9).

English sentence: *"*a lot of their books*"* …(6.9)

UNL of (6.9) is given in (6.10).

{unl}
pos(book:01.@multal, 00:02.@3.@pl)
{/unl} …(6.10)

Expected Punjabi sentence: ਉਹਨਾਂ ਦੀਆਂ ਕਈ ਕਿਤਾਬਾਂ …(6.11)

Transliterated Punjabi sentence: *uhnan dian kai kitaban*

All nodes (or UWs) identified in UNL sentence given in (6.10), are mapped with their corresponding dictionary entries, and are listed below.

00:02.@3.@pl - [ਉਹ]{}"00.@3.@pl" (LEX=R,POS=POD,PAR=M7);

book:01.@multal-[ਕਿਤਾਬ]{}"book"(LEX=N,POS=NOU,NUM=SNG,GEN=FEM, PAR=M1);

The process of NLization of example UNL sentence given in (6.10) has been illustrated in Table 6.3. Transliteration of each Punjabi word is also mentioned in the table.

Table 6.3: NLization Process of Example Sentence (6.10)

Rule Fired	Description	Action Taken
(%y,M1):=(%y,+FLX(SNG:=0> "";PLR:=0> "ਆਂ");),-M1);	This paradigm M1 attaches the corresponding postfix to nouns in Punjabi. The attribute "FLX" indicates that node will be handled for inflection according to the given rule.	To: [book:01.@indef] Result: ["ਕਿਤਾਬ":01.@indef] *["kitab":01.@indef]* As the UW "ਕਿਤਾਬ" "*kitab*" does not have any combination of attributes mentioned in the given rule so, nothing is appended to this word.
(%x,M7,+FLX(SNG&BEN &PAS:=1>"ਸਨੂੰ"; SNG&AGT &PAS:=1>"ਸਨੇ";SNG&POS &MCL&^DET:=1>"ਸਦਾ"; SNG&POS&FEM&^DET:=1> "ਸਦੀ"; PLR&POS&MCL&	This paradigm M7 attaches corresponding postfix to pronoun "ਉਹ" "*uh*" in Punjabi. The attribute "FLX" indicates that node will be handled	To: [00:02.@3.@pl] Result: ["ਉਹ":02.@3.@pl] *["uh":02.@3.@pl]*

^DET:=1>"**ਸਦੇ"**; PLR&POS &FEM&^DET:=1>"**ਸਦੀਆਂ"**; PLR&BEN&PAS&DET:=0> "**ਨਾਂਨੂੰ"**;PLR&POS&DET:=0>"**ਨਾਂ"** ;**))**	for inflection according to the given rule.	As the UW "**ਉਹ**" "*uh"* does not have any combination of attributes mentioned in the given rule so, nothing is appended to this word.
pos(%x,N,FEM,@multal;%y, @pl,@3,POD):=(%y,+PLR, +FEM, +POS,+DET,-@pl) ("**ਦੀਆਂ**") **(%**x);	This T-rule has been defined to resolve the "pos" relation between noun and pronoun. It places pronoun before the noun and inserts a special word "**ਦੀਆਂ**" "*dian"* between them. Also, it removes "@pl" and adds "PLR", "POS", "DET" attributes from second UW.	To: pos(book**:**01.multal ;%b, 00.@3.@pl) Result: #L(00.@3, **ਦੀਆਂ** :04); #L(**ਦੀਆਂ** :04 , book**:**01. @multal) *#L(00.@3, dian:04);* *#L(dian :04, book**:**01. @multal)* Intermediate Result: **ਉਹ ਦੀਆਂ ਕਿਤਾਬ** *uh dian kitab* "pos" relation gets resolved by placing pronoun before the noun and inflected word "**ਦੀਆਂ**" "*dian"* get inserted between them.
(%x,N,SNG,@multal):=("**ਕਈ**") (" ")(%x,-@multal,-NUM, +NUM=PLR);	This T-rule has been defined to remove an attribute "@multal" from	To:["**ਕਿਤਾਬ**"**:**01.@multa l] Result: #L(**ਕਈ:**01, -

	a noun, add a special word "ਕਈ" "*kai*" before it and update the number information of noun to "PLR".	:02); #L(-:02, "ਕਿਤਾਬ":01) *#L(kai:01, -:02);* *#L(-:02, "kitab":01)* Intermediate Result: ਉਹ ਦੀਆਂ ਕਈ ਕਿਤਾਬ *uh dian kai kitab* Attribute "@multal" gets removed and a special word "ਕਈ" "*kai"* is added before noun "ਕਿਤਾਬ" "*kitab"*.
({D|N|R|V|J}, ^inflected, FLX, %y):=(!FLX,+inflected,-FLX, %y);	It fires the corresponding inflection paradigm to inflect the root word "ਉਹ" uh to "ਉਹਨਾਂ" uhnan based on the combination "PLR&POS&DET" of attributes it has.	To: [00:02.@3.@pl] Result: ["ਉਹਨਾਂ":02.@3] *["uhnan":02.@3]* Final output: ਉਹਨਾਂ ਦੀਆਂ ਕਈ ਕਿਤਾਬ *uhnan dian kai kitab* Attribute "PLR" was associated with "ਕਿਤਾਬ" "*kitab"*, thus, it gets inflected to "ਕਿਤਾਬਾਂ" "*kitaban"*

({D|N|R|V|J}, ^inflected, FLX, %y):=(!FLX,+inflected,-FLX, %y);	It fires the corresponding paradigm rule to inflect the root word "ਕਿਤਾਬ" "*kitab"* to make it "ਕਿਤਾਬਾਂ" "*kitaban"* based on the number information of the word.	To: ["ਕਿਤਾਬ":01] Result: ["ਕਿਤਾਬਾਂ":01] Final output: ਉਹਨਾਂ ਦੀਆਂ ਕਈ ਕਿਤਾਬਾਂ *uhnan dian kai kitaban* Attribute "PLR" was associated with "ਕਿਤਾਬ" "*kitab"*, thus, it gets inflected to "ਕਿਤਾਬਾਂ" "*kitaban"*

The natural language output generated by EUGENE for sentence (6.9) is given in (6.12), correspond to UNL sentence given in (6.10).

{org}
a lot of their books
{/org}
{pun}
ਉਹਨਾਂ ਦੀਆਂ ਕਈ ਕਿਤਾਬਾਂ
{/pun}
{unl}
[W]
pos(book:01.@multal, 00:02.@3.@pl)
[/W]
{/unl} …(6.12)

6.2.4 NLization of Verbs

The NLization of verbs in a UNL sentence to NL sentence is shown using the following example given in (6.13).

English sentence: "He will not arrive" …(6.13)

UNL of (6.13) is given in (6.14).

{unl}

agt(arrive:03.@not.@future, 00:01.@3.@male)

{/unl} …(6.14)

Expected Punjabi sentence: ਉਹ ਨਹੀਂ ਪਹੁੰਚੇਗਾ …(6.15)

Transliterated Punjabi sentence: *uh nahin phunchega*

All nodes (or UWs) identified in UNL sentence given in (6.14), are mapped with their corresponding dictionary entries, and are listed below.

"00:01" - [**ਉਹ**]{}"00.@3.@male" (LEX=D,POS=POD,PAR=M7);

arrive:03.@not.@future-[**ਪਹੁੰਚ**]{}"arrive"(LEX=V,POS=VER,TRA=TSTI,PAR=M4);

The process of NLization of example UNL sentence given in (6.14) has been illustrated in Table 6.4. The transliteration of each Punjabi word is also mentioned in the table.

Table 6.4: NLization Process of Example Sentence (6.14)

Rule Fired	Description	Action Taken
(%y,M4):=(%y,-M4,+FLX(MCL&PST&SNG&^ANT&^PGS :=0>" **ਗਿਆ ਸੀ**";PST&SNG&FEM &^PGS& ^ANT:=0>" **ਗਈ ਸੀ**"; PST &PLR&FEM&^ANT:=0>" **ਗਈਆਂ ਸਨ**" ;MCL&PLR&PST :=0>" **ਗਏ ਸਨ**" ; SNG&PRS &MCL&^PER &^PGS:=0>"**ਦਾ ਹੈ**"; PRS&SNG& FEM&^PGS &^PER:=0> "**ਦੀ ਹੈ**"; PRS&PLR&FEM:=0> "**ਦਿਆ ਹਨ**"; PRS&PLR&MCL :=0>" **ਦੇ ਹਨ**"; PRS&SNG&MCL&PGS:=0>	This paradigm attaches corresponding postfix to verbs in Punjabi. The attribute "FLX" indicates that node will be handled for inflection as per this given rule.	To: [arrive:03.@future. @not] Result: ["**ਪਹੁੰਚ**"**:**03.@future. @not] *["phunch"**:**03.@future. @not]* As the UW "**ਪਹੁੰਚ**" "*phunch"* does not have any combination of attributes mentioned in the given rule so, nothing is appended to this word.

"ਰਿਹਾ ਹੈ"; PRS&SNG&FEM &PGS:=0>" ਰਹੀ ਹੈ"; PRS&PLR&FEM&PGS:=0>" ਰਹੀਆਂ ਹਨ"; PRS&PLR&MCL&PGS:=0> "ਰਹੇ ਹਨ"; PST&SNG&MCL&PGS:=0> "ਰਿਹਾ ਸੀ"; PST&SNG&FEM&PGS:=0> "ਰਹੀ ਸੀ"; PST&PLR&FEM&PGS:=0> "ਰਹੀਆਂ ਸਨ"; PST&PLR&MCL&PGS:=0> "ਰਹੇ ਸਨ"; {PST&MCL&SNG&ANT}:=0> "ਚੁੱਕਾ ਸੀ"; {PST&FEM&SNG&ANT}:=0> "ਚੁੱਕੀ ਸੀ"; {PST&MCL&PLR&ANT}:=0> "ਚੁੱਕੇ ਸਨ"; {PST&FEM&PLR&ANT}:=0> "ਚੁੱਕੀਆਂ ਸਨ"; {PER&PRS&MCL&SNG}:=0> "ਚੁੱਕਾ ਹੈ"; {PER&PRS&FEM&SNG}:=0> "ਚੁੱਕੀ ਹੈ"; {PER&FUT&MCL&SNG}:=0> "ਚੁੱਕਾ ਹੋਵੇਗਾ"; {PER&FUT&FEM&SNG}		

:=0>"ਚੁੱਕੀ ਹੋਵੇਗੀ"; {FUT&MCL&PGS&SNG}:=0> "ਰਿਹਾ ਹੋਵੇਗਾ"; {FUT&FEM&PGS&SNG}:=0> "ਰਿਹਾ ਹੋਵੇਗੀ"; {FUT&MCL&SNG }:=0>"ੇਗਾ"; FUT &FEM&SNG&^PGS &^PER&^RES:=0>"ੇਗੀ";));		
(%x,M7,+FLX(SNG&BEN&PAS :=1>"ਸਨੂੰ"; SNG&AGT&PAS:=1>"ਸਨੇ"; SNG&POS&MCL&^DET:=1> "ਸਦਾ"; SNG&POS&FEM&^DET:=1> "ਸਦੀ"; PLR &POS&MCL&^DET:=1> "ਸਦੇ"; PLR&POS& FEM&^DET:=1> "ਸਦੀਆਂ"; PLR&BEN&PAS&DET:=0> "ਨਾਂ ਨੂੰ"; PLR&POS&DET:=0>"ਨਾਂ";))	This paradigm M7 has been defined to attach corresponding postfix to pronoun "ਉਹ" "*uh"* in Punjabi. The attribute "FLX" indicates that the node will be handled for inflection as per this given rule.	To: [00:03.@3.@pl] Result: ["ਉਹ":03.@3.@pl] ["*uh*":03.@3.@pl] As the UW "ਉਹ" "*uh*" does not have any combination of attributes mentioned in the given rule so, nothing is appended to this word.
agt(%a,V,@future,^FUT;%b,R):= agt(%a ,+FUT,-@future;%b) ;	This rule adds the attribute "FUT" to the verb node on the basis of the attributes associated with it because "FUT" combination is important to append correct postfix to	To:agt(arrive:03.@not, 00:01.@3.@male) Result: agt(arrive:03,00:01.@3) Nothing is appended to any Punjabi root word as postfix and

	Punjabi root word verb.	“@future” attribute is removed from second node “%b” and “FUT” is added to first node “%a”.
agt(%a,^MCL;%b,R,@male):= agt(%a ,+MCL,-NUM,+NUM =SNG; %b,-@male);	This rule adds the attributes “MCL” and “SNG” to the verb on the basis of the attribute “@male” that is associated with the node “%b” because combination of “FUT&MCL&SNG” is required to append correct postfix to Punjabi root word.	To:agt(arrive:03.@not , 00:01.@3.@male) Result: agt(arrive:03.@not, 00:01.@3) Nothing is appended to any Punjabi root word as postfix, just “@male” attribute is removed from node “%b” and the attributes “MCL” and “SNG” are added to node “%a”.
agt(%a,V;%b,R):=(%b)(" ")(%a);	It resolves the “agt” relation, places pronoun before verb and introduces new node of single space (“ ”) between two nodes.	To: agt(arrive:03.@not, 00:01.@3) Result: #L(00:01.@3, -:02); #L(-:02, arrive:03.@not) Intermediate Result: ਉਹ ਪਹੁੰਚ *uh phunch* As a result, “agt”

		relation is resolved by placing the pronoun "ਉਹ" "*uh"* before verb "ਪਹੁੰਚ" "*phunch"*.
(%x,V,@not):=)(" ਨਹੀਂ ")(%x, -@not);	This T-rule has been defined to remove an attribute "@not" from a verb, add a negation word "ਨਹੀਂ" "*nahi"* before it.	To: [arrive:03.@not] Result: #L(00:01.@3, -:04); #L(-:04 ਨਹੀਂ :05, arrive:03) *#L(-:04 nahin :05, arrive:03)* Intermediate Result: ਉਹ ਨਹੀਂ ਪਹੁੰਚ *uh nahin phunch* Attribute "@not" gets removed and a negation word, "ਨਹੀਂ" "*nahi"* is added before verb "ਪਹੁੰਚ" "*phunch"*.
({N|V|D|J|R},FLX,^inflected,%x):=(!FLX,-FLX,+inflected,%x);	This rule fires the corresponding paradigm rule to inflect the root word "ਉਹ" "*uh"*.	To:[00:01.@3] Result:["ਉਹ":01.@3] *["uh":01.@3]* As the UW "ਉਹ" "*uh*" does not have any combination of attributes mentioned in

		the given rule so, nothing is appended to this word.
{N|V|D|J|R},FLX,^inflected,%x): =(!FLX,-FLX,+inflected,%x);	This rule fires the corresponding paradigm rule to inflect the root word "ਪਹੁੰਚ" "*phunch"* to make it "ਪਹੁੰਚੇਗਾ" *"phunchega"*	To:[arrive:03] Result:[" ਪਹੁੰਚੇਗਾ "**:**07] *["phunchega "**:**07]* Final Output: ਉਹ ਨਹੀਂ ਪਹੁੰਚੇਗਾ *uh nahin phunchega* The root word "ਪਹੁੰਚ" "*phunch"* get inflected to "ਪਹੁੰਚੇਗਾ" "*phunchega"*.

The natural language output generated by EUGENE for sentence (6.13) is given in (6.16), correspond to UNL sentence given in (6.14).

{org}
He will not arrive
{/org}
{pun}ਉਹ ਨਹੀਂ ਪਹੁੰਚੇਗਾ {/pun}
{unl}
[W] agt(arrive:03.@not.@future, 00:01.@3.@male) [/W]
{/unl} …(6.16)

6.2.5 NLization of Nouns

The NLization of nouns in a UNL sentence to NL sentence is shown using the following example given in (6.17).

English sentence: "All the books available" …(6.17)

UNL of (6.17) is given in (6.18).

{unl}
mod(book:05.@def.@all,available:07)

{/unl} …(6.18)

Expected Punjabi sentence: ਸਾਰੀਆਂ ਉਪਲਬਧ ਕਿਤਾਬਾਂ

Transliterated Punjabi sentence: *saarian uplabadh kitabaan*

After the tokenization of example sentence given in (6.18) with EUGENE tool, two lexical items are identified as shown below.

[ਉਪਲਬਧ]{}"available"(LEX=J,POS=ADJ)<pun,0,0>;

[ਕਿਤਾਬ]{}"book"(LEX=N,POS=NOU,NUM=SNG,GEN=FEM,PAR=M1)<pun,0,0>;

Here, "*LEX"* represents a lexical category, "*J*" represents adjective, "*POS*" represents part-of-speech, its value could be either "*ADJ*" or "*NOU*", "*ADJ*" indicates adjective, "*NOU*" indicates a common noun. "*GEN*" represents gender whose value could be either "*MCL*" for male or "*FEM*" for female, "*NUM*" represents number whose value could be either "*SNG*" for singular or "*PLR*" for plural, *"PAR"* represents paradigm which is used to generate the inflected forms out of the base form. The process of NLization of example sentence given in (6.18) has been illustrated in Table 6.5.

Table 6.5: NLization of Example Sentence (6.18)

S.No	Rule Fired	Description	Action Performed
1.	mod(%a,N,@all;%b,J): =((%b)(" ")(%a,+NUM =PLR ,-@all),+@all, +LEX= %a, +NUM= %a,+GEN=%a);	This rule resolves the *"mod"*, *i.e.*, modification relation between the nodes. It creates two nodes and places node "%b" before node "%a".	**UW View:** L:01(available:07, -:02) #L:01(-:02, book:05.@def) **String View:** #L:01("ਉਪਲਬਧ":07, " " :02) #L:01(" ":02, "ਕਿਤਾਬ":05. @def) *#L:01("uplabadh":07, " " :02)* *#L:01(" ":02, "kitab":05. @def)*

			Generated output: sc:01{[“ ਉਪਲਬਧ”] [" "] ["ਕਿਤਾਬ"]} *sc:01{["uplabadh "][" "]["kitab"]}* The relation *“mod”* is resolved; “available” is placed before “book” with [“ ”] as blank space between them. Node: 01 indicates scope node internally generated by EUGENE.
2.	(N,SNG,FEM,@all,%a):=("ਸਾਰੀਆਂ")(" ")(%a,-@all, -NUM,+NUM =PLR);	This rule resolves the attribute “@all” to insert word "ਸਾਰੀਆਂ" before the node “%a”, which is having a lexical category noun and number value singular.	**UW View:** #L:01(available:07, -:02) #L:01(-:02, book:05.@def) **String View:** #L("ਸਾਰੀਆਂ":03, " ":04) #L(" ":04, :01) #L:01("ਉਪਲਬਧ":07, " " :02) #L:01(" ":02, "ਕਿਤਾਬ":05. @def) *#L("sariaan":03, " ":04)* *#L(" ":04, :01)*

			#L:01("uplabadh":07, " " :02) *#L:01(" ":02, "kitab":05. @def)* **Generated output:** ["ਸਾਰੀਆਂ"][" "]["ਉਪਲਬਧ"] [" "] ["ਕਿਤਾਬ"] *["sariaan"][" "]* *["uplabadh"][" "]* *["kitab"]* As a result, "@all" attribute is resolved, and blank space [" "] is inserted between UWs.
3.	(%a,@def):=(%a,-@def);	This rule removes the attribute "@def" from the node "book".	**UW View:** #L(-:04, :01) #L:01(available:07, -:02) #L:01(-:02, book:05.@def) **String View:** #L("ਸਾਰੀਆਂ":03, " ":04) #L(" ":04, :01) #L:01("ਉਪਲਬਧ":07, " " :02) #L:01(" ":02, "ਕਿਤਾਬ":05) *#L("sariaan":03, " ":04)* *#L(" ":04, :01)*

			#L:01("uplabadh":07, " " :02) *#L:01(" ":02, "kitab":05)* **Generated output:** ["ਸਾਰੀਆਂ"][" "]["ਉਪਲਬਧ"] [" "] ["ਕਿਤਾਬ"] *["sariaan"][" "] ["uplabadh"] [" "] ["kitab"]* As a result, "@def" attribute is removed from the UW "book".
4.	(%x,M1):=(%x,-M1,+FLX (SNG:=0>"";PLR:=0>"ਾਂ";));	This paradigm M2 attaches word "ਾਂ" with node "%x" if it is plural, otherwise, it remains same.	No action performed
5.	({N|V|D},FLX, ^inflected,%x):=(!FLX,-FLX, +inflected,%x);	This rule fires the corresponding paradigm rule M2 to inflect the root word "book".	**Generated output:** ["ਸਾਰੀਆਂ"][" "]["ਉਪਲਬਧ"] [" "] ["ਕਿਤਾਬਾਂ"] *["sariaan"][" "] ["uplabadh"][" "] ["kitabaan"]* There is a change in the UW "book" because it is plural.

The natural language output generated by EUGENE for sentence (6.17) is given in (6.19), corresponding to the UNL sentence given in (6.18).

{org}
All the books available
{/org}
{pun}
ਸਾਰੀਆਂ ਉਪਲਬਧ ਕਿਤਾਬਾਂ
{/pun}
{unl}
[W] pos(car:04, John:01) [/W]
{/unl} …(6.19)

6.2.6 NLization of Pronouns

The NLization of pronouns in a UNL sentence to NL sentence is shown using the example given in (6.20).

English sentence: “A book of mine” …(6.20)

UNL of (6.20) is given in (6.21).

{unl}
pos (book:03.@indef, 00:07.@1)
{/unl} …(6.21)

Expected Punjabi sentence: ਮੇਰੀ ਇਕ ਕਿਤਾਬ

Transliterated Punjabi sentence: *meri ik kitaab*

After the tokenization of example sentence given in (6.21) with EUGENE tool, two lexical items are identified as shown below.

[ਮੇਰੀ]{}"00.@1" (LEX=D,POS=POD,PAR=M3)<eng,255,0>;
[ਕਿਤਾਬ]{}"book"(LEX=N,POS=NOU,NUM=SNG,GEN=FEM,PAR=M1)<pun,0,0>;

Here, “*LEX*” represents lexical category, “*D*” represents determiner, “*POS*” represents part-of-speech whose value is “*POD*” which indicates possessive determiner. “*GEN*” represents gender value whose value could be either “*MCL”* for male or “*FEM*” for female, “*NUM*” represents number whose value could be either “*SNG*” for singular or “*PLR*” for plural, “*PAR*” represents paradigm which is used to generate the inflected

forms out of the base form. The process of NLization of example sentence (6.21) has been illustrated in Table 6.6.

Table 6.6: NLization of Example Sentence (6.21)

S.No.	Rule Fired	Description	Action Performed
1.	(%y,M1):=(%y,-M1, +FLX(SNG:=0>""; PLR:=0>"ਾਂ";));	This paradigm attaches word "ਾਂ" with node "%x" if it is plural, otherwise it remains same.	No action performed
2.	(%x,M3):=(%x,-M3, +FLX(SNG:=0>""; PLR:=0>"ਆਂ";MCL: =1>"ਾ"));	This paradigm attaches word "ਆਂ" with node "%x" if it is plural, otherwise it remains same.	No action performed
3.	({N|V|D|J},FLX,^inflected,%x):=(!FLX,-FLX,+inflected,%x);	This rule fires the corresponding paradigms M1 and M3 to inflect the root word.	There is no change in the UW "00:07.@1" and "book" because these are singular.
4.	pos(%a,N;%b,POD):=(%b)("") (%a);	This rule places the node "%b" before node "%a". "pos" indicates the possessor of a thing where UW2 is the possessor of UW1.	UW View: #L(00:07.@1,-:01) #L(-:01,book:03. @indef) String View: #L("ਮੇਰੀ":07.@1," ":01) #L(" ":01,"ਕਿਤਾਬ":03. @indef) *#L("meri":07.@1," ":01)*

			#L(" ":01,"kitab":03. @indef) Generated output: ["ਮੇਰੀ"] [" "] ["ਕਿਤਾਬ"] *["meri"] [" "] ["kitab"]* As a result, *"pos"* relation is resolved; UW "00:07.@1" is placed before "book". [" "] indicates a blank space between UWs.
5.	(N,@indef,%a):=(("ਇਕ")("")(%a,- @indef), %a,+NA, -@indef);	This rule inserts the word "ਇਕ" *"ik"* before node "%a" which is a noun and it also resolves the attribute "@indef".	UW View: #L(00:07.@1, -:01) #L(-:01, :01) #L:01(ਇਕ:02, -:04) *#L:01(ik:02, -:04)* #L:01(-:04, book:03) String View: #L("ਮੇਰੀ":07.@1,"":01) *#L("meri":07.@1,"":01)* #L(" ":01, :01) #L:01("ਇਕ":02,"":04) #L:01(" ":04,"ਕਿਤਾਬ" :03) *#L:01("ik":02,"":04)* *#L:01(" ":04,"kitab" :03)* Generated output:

			["ਮੇਰੀ"] [" "] ["ਇਕ"] [" "] ["ਕਿਤਾਬ"] *["meri"] [" "] ["ik"] [" "] ["kitab"]* As a result, "@indef" attribute is resolved; "00:07.@1" is placed before "book" with "ਇਕ" between them.

The natural language output generated by EUGENE for sentence (6.20) is given in (6.22), corresponding to UNL sentence given in (6.21).

{org}
a book of mine
{/org}
{pun}
ਮੇਰੀ ਇਕ ਕਿਤਾਬ
{/pun}
{unl}
[W] pos(book:03.@indef, 00:07.@1) [/W]
{/unl} …(6.22)

6.2.7 NLization of Prepositions

The NLization of prepositions in a UNL sentence to NL sentence is shown using the example given in (6.23).

English sentence: *"The train from Rome to Paris through Geneva"* …(6.23)

UNL of (6.23) is given in (6.24) and its UNL graph is shown in Figure 6.8.

{unl}
plc(train:03.@def, Rome:07.@from)
plc(train:03.@def, Paris:0B.@to)
plc(train:03.@def, Geneva:0F.@through)
{/unl} …(6.24)

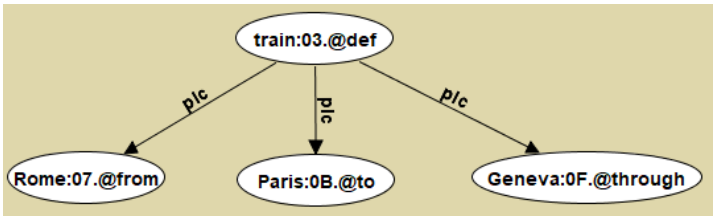


Figure 6.8: UNL Graph of Example (6.23)

Expected Punjabi sentence: ਰੋਮ ਤੋਂ ਪੈਰਿਸ ਨੂੰ ਜਨੇਵਾ ਵਿਚੋਂ ਰੇਲ ਗੱਡੀ

Transliterated Punjabi sentence: *rom ton pairis nu geneva vichon rel gaddi*

After the tokenization of example sentence given in (6.24) with EUGENE tool, four lexical items are identified as shown below.

[ਪੈਰਿਸ]{}"Paris"(LEX=N,POS=PPN,NUM=SNGT,PAR=M0)<pun,0,0>;

[ਰੇਲ ਗੱਡੀ]{}"train" (LEX=N,POS=NOU,NUM=SNG)<pun,0,0>;

[ਰੋਮ]{}"Rome"(LEX=N,POS=PPN,NUM=SNGT,PAR=M0)<pun,0,0>;

[ਜਨੇਵਾ]{}"Geneva"(LEX=N,POS=PPN,NUM=SNGT)<pun,0,0>;

Here, "*LEX"* represents a lexical category, "*D*" represents determiner, "*POS*" represents part-of-speech whose value is "*POD*" which indicates possessive determiner. "*GEN*" represents gender value whose value could be either "*MCL*" for male or "*FEM*" for female, "*NUM*" represents number whose value could be either "*SNG*" for singular or "*PLR*" for plural, "*PAR*" represents paradigm which is used to generate the inflected forms out of the base form. The process of NLization of example sentence (6.24) has been illustrated in Table 6.7.

Table 6.7: NLization of Example Sentence (6.24)

S.No.	Rule Fired	Description	Action Performed
1.	(%y,@def):=(%y,-@def);	This rule removes the attribute "@def" from the node.	UW View: plc(train:03.@def,Rome:07.@from) plc(train:03.@def, Paris:0B.@to) plc(train:03.@def, Geneva:0F.@through) String View:

			plc("ਰੇਲ ਗੱਡੀ":03, "ਰੋਮ": 07.@from) plc("ਰੇਲ ਗੱਡੀ":03, "ਪੈਰਿਸ" :0B.@to) plc("ਰੇਲ ਗੱਡੀ":03, "ਜਨੇਵਾ" :0F.@through) *plc("rel gaddi":03, "rom": 07.@from)* *plc("rel gaddi":03, "paris" :0B.@to)* *plc("rel gaddi":03, "geneva" :0F.@through)* As a result, "@def" attribute is resolved with UW "train".
2.	(N,PPN,SNGT,@from ,%a):=((%a,-@from) ("ਤੋਂ"),N,PPN,SNGT, DONE,%f);	This rule resolves "@from" attribute to insert word "ਤੋਂ" "*ton"* after the node "%a", which is having a lexical category noun and part of speech as proper noun.	UW View: #L:01(Rome:07, ਤੋਂ :02) String View: #L:01("ਰੋਮ":07, "ਤੋਂ":02) *#L:01("rom":07, "ton":02)* Generated output:

			sc:01{ ["ਰੋਮ"] ["ਤੋਂ"] } *sc:01{ ["rom"] ["ton"] }* As a result, "@from" attribute is resolved. UW "Rome" is placed before word "ਤੋਂ" "*ton"*.
3.	(N,PPN,SNGT,@to, %a):=((%a,@to)("ਨੂੰ"), N,PPN,SNGT,DONE, %f);	This rule resolves "@to" attribute to insert word "ਨੂੰ" "*nu"* after the node "%a" which is having a lexical category noun and part of speech as proper noun.	UW View: #L:02(Paris:0B, ਨੂੰ:05) *#L:02(Paris:0B, nu :05)* String View: #L:02("ਪੈਰਿਸ":0B,"ਨੂੰ":05) *#L:02("paris":0B,"nu":05)* Generated output: sc:02{["ਪੈਰਿਸ"] ["ਨੂੰ"]} *sc:02{["paris"] ["nu"]}* As a result, "@to" attribute is resolved. UW "Paris" is placed before word "ਨੂੰ" "*nu"*.
4.	(N,PPN,SNGT, @through,%a):=((%a, @through)("ਵਿਚੋਂ"),N, PPN,SNGT,DONE, %f);	This rule resolves "@through" attribute to insert word "ਵਿਚੋਂ" *"vichon"* after the node "%a" which is having a lexical category noun and part of speech as	UW View: #L:03(Geneva:0F,ਵਿਚੋਂ :08) *#L:03(Geneva:0F,vichon :08)* String View:

		proper noun.	#L:03("ਜਨੇਵਾ":0F,"ਵਿਚੋਂ" :08) *#L:03("geneva":0F,"vichon" :08)* Generated output: sc:03{ ["ਜਨੇਵਾ"]["ਵਿਚੋਂ"]} *sc:03{["geneva"] ["vichon]}* As a result, "@through" attribute is resolved. UW "Geneva" is placed before word "ਵਿਚੋਂ" "*vichon"*.
5.	plc(N,NOU,%a;N,{ PPN|NOU},%b):=((%b)(%a),%f);	This rule resolves the "plc", *i.e.*, place relation between the nodes. It creates two nodes and places node "%b" before node "%a". The relation place specifies a place where an event occurs. Here, UW1 is a state, place or an event and UW2 is a thing/place taken as a place.	UW View: #L:04(:01, :02) #L:01(Rome:07, ੋ :02) *#L:01(Rome:07, ton:02)* #L:02(Paris:0B, ੂ :05) *#L:02(Paris:0B, nu :05)* #L:06(:03, train:03) #L:03(Geneva:0F, ਵਿਚੋਂ :08) *#L:03(Geneva:0F, vichon: 08)* String View: #L:04(:01, :02)

			#L:01("ਰੋਮ":07,"ਤੋਂ":02) #L:02("ਪੈਰਿਸ":0B,"ਨੂੰ":05) #L:06(:03,"ਰੇਲ ਗੱਡੀ":03) #L:03("ਜਨੇਵਾ":0F,"ਵਿਚੋਂ ":08) *#L:04(:01, :02)* *#L:01("rom":07,"ton":02)* *#L:02("paris":0B,"nu":05)* *#L:06(:03,"rel gaddi":03)* *#L:03("geneva":0F,"vichon ":08)* Generated output: sc:04{ ["ਰੋਮ"] ["ਤੋਂ"] ["ਪੈਰਿਸ"] ["ਨੂੰ"] ["ਜਨੇਵਾ"] ["ਵਿਚੋਂ"] ["ਰੇਲ ਗੱਡੀ"] *sc:04{ ["rom"] ["ton"] ["paris"] ["nu"] ["geneva"] ["vichon"] ["rel gaddi"]* As a result "plc" relation is resolved; and the output is generated.

The natural language output generated by EUGENE for ssentence (6.23) is given in (6.25), corresponding to the UNL sentence given in (6.24).

{org}

the train from Rome to Paris through Geneva

{/org}

{pun}

ਰੋਮ ਤੋਂ ਪੈਰਿਸ ਨੂੰ ਜਨੇਵਾ ਵਿਚੋਂ ਰੇਲ ਗੱਡੀ

{/pun}

{unl}

plc(train:03.@def, Rome:07.@from)

plc(train:03.@def, Paris:0B.@to)

plc(train:03.@def, Geneva:0F.@through)

{/unl} …(6.25)

The different cases discussed above show the working of EUGENE tool for conversion of input UNL sentence to Punjabi language. Now this EUGENE/ NLization module (configured with generation dictionary and TRules) is invoked by the generation module of the developed QA system to get the answer in the target natural language.

6.3 Generation Module (NLization Phase)

The generation module of the developed QA system invokes EUGENE to NLize the output UNL. For NLization process, generation dictionaries, TRules, and DRules are created by the computational linguists of the respective natural language and stored at UNL web. These resources are created according to UNL specifications [153]. The generation module of the developed QA system converts the output of optimizer to the target natural language.

The answer keyword generated by optimizer is converted to UNL and given to the generation module of the developed QA system for converting it to the desired target natural language by invoking its EUGENE module.

Let us reconsider the previous example given in (5.4) of Chapter 5 in (6.26) and observe the output of this example given in Figure 6.9.

ਆਧੁਨਿਕ ਖੇਤੀਬਾੜੀ ਕਿਹੜੀ ਚੀਜ਼ ਉੱਪਰ ਨਿਰਭਰ ਹੈ?

Ādhunika khētībāṛī kihaṛī cīza upara nirabhara hai?

“On what does modern agriculture depends?” …(6.26)

For this example sentence, the generation module of the “Punjabi” natural language is called and the output given by the UNL crawler is NLized and answer is given in target natural language (*i.e.*, Punjabi) as shown in Figure 6.9.

Similar to analysis module (which invokes IAN/UNLization module), generation module invokes EUGENE/ NLization of the target natural language. It calls an API through the developed interface which requires the following parameters:

a. User name
b. Web service client name
c. Three character EUGENE ISO code
d. Language name
e. Output given by the optimizer

The values of all these parameters change with the language selection. Once the call to an API is made, it returns the natural language text successfully.

If EUGENE is not able to NLize the keywords, then that keyword is returned as it is. Figure 6.9 shows the final answer in the desired target language.

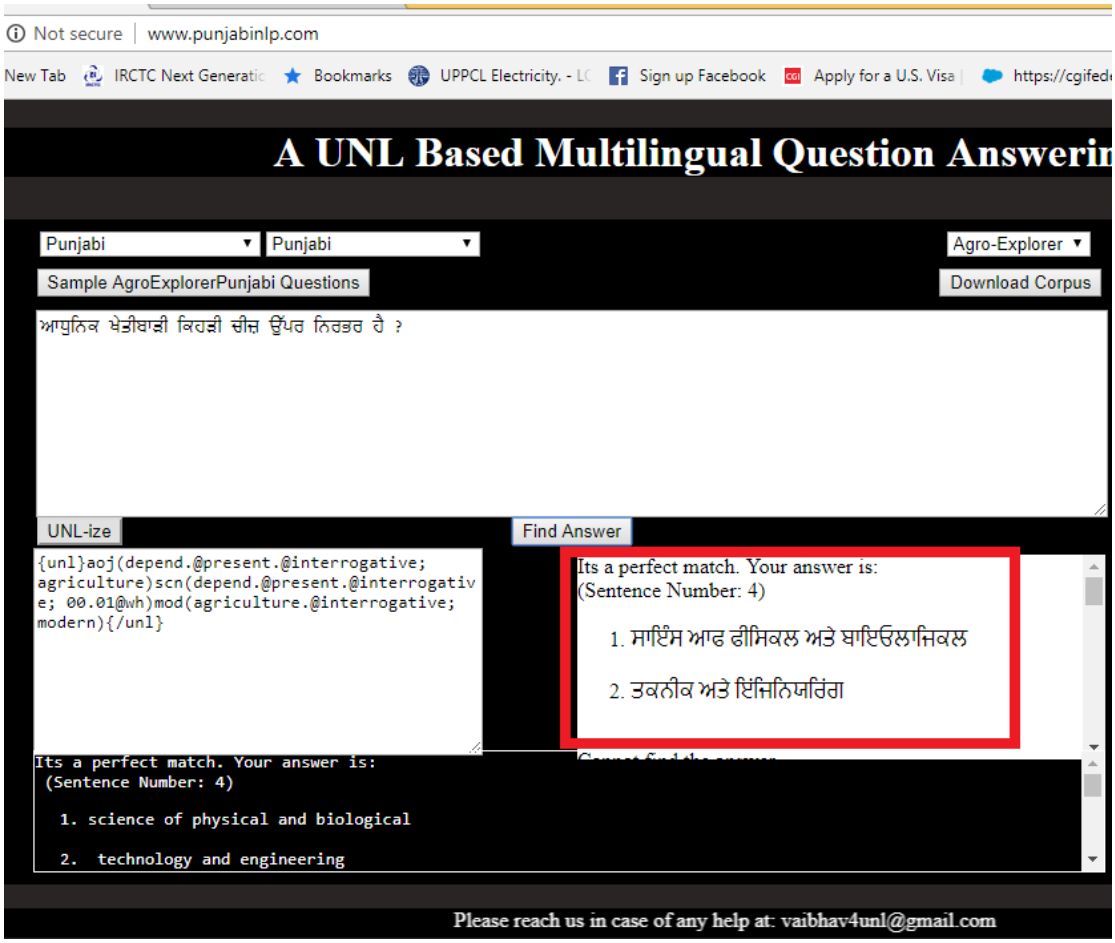


Figure 6.9: Output Given by the Generation Module

6.4 State of Art for EUGENE/NLization Module of Punjabi Language

Earlier work of NLization of adjectives, conjunctions, determiners, verbs, nouns, pronouns, and prepositions with EUGENE for the Punjabi language has helped us to configure EUGENE tool for proposed QA system [135][164][165].

The work presented in this chapter for the proposed QA System, *i.e.*, Punjabi NLization module, has been selected in top 5 (based upon the F-Measures) NLization grammars for Olympiad III, and Olympiad IV conducted by UNDL foundation, Geneva. F-Measure of Punjabi language NLization and other best-selected languages for Olympiad III, and Olympiad IV is shown in Figure 6.10 and Figure 6.11 respectively. The highlighted entries in Figure 6.10 and Figure 6.11 refer to Punjabi language results.

NLization

General Position	Language Pair	Author	F-Measure	Submission Date	Medal	Files			
						Dictionary	T-Grammar	D-Grammar	Output
1	unl>slv	Grega Milharcic	1.000	07/03/2014	GOLD	[21]	[22]	[23]	[24]
2	unl>bul	Yordanka Stancheva	1.000	21/03/2014	GOLD	[25]	[26]	[27]	[28]
3	unl>pan	Parteek Kumar	0.996	30/03/2014	GOLD	[29]	[30]	[31]	[32]
4	unl>per	Maryam Faal Hamedanchi	0.994	29/03/2014	GOLD	[33]	[34]	[35]	[36]
5	unl>ukr	Sergiy Prots	0.957	30/03/2014	GOLD	[37]	[38]	[39]	[40]
6	unl>rus	Sergiy Prots	0.916	29/03/2014	GOLD	[41]	[42]	[43]	[44]

Figure 6.10: F-Measure for Top 6 NLization Best-Selected Languages for Olympiad III [150]

NLization (Qualified Grammars Only)

Submission	Author	L Pair	F-Measure	Medal	Position	Files				
						Corpus	Dic	T-grammar	D-grammar	Output
11-11-2014	Parteek Kumar	unl>pan	1.000	GOLDEN	1	[31]	[32]	[33]	[34]	[35]
15-11-2014	Parteek Kumar	unl>hin	1.000	GOLDEN	2	[36]	[37]	[38]	[39]	[40]

Figure 6.11: F-Measure for Top 2 Best NLization Selected Languages for Olympiad IV [49]

The corpus used in NLization Olympiad III and Olympiad IV were same which were used for UNLization Olympiad III and IV, *i.e.*, UGO-A1 and AEOP-A1 respectively .

As mentioned earlier, the work has been merged and extended for the proposed QA system and the current updated NLization/EUGENE module for the Punjabi language has 704 UWs, 462 TRules, and 10 inflectional paradigms.

Table 6.8 shows the details of EUGENE/NLization module for NLization of Punjabi language.

Table 6.8: Details of EUGENE/NLization Module for NLization of Punjabi language

NLization Assets	**Count/ Value**
TRules	462
Dictionary entries	704
Inflectional paradigms	10

Chapter Summary

In this chapter, the detailed information of generation, NLization/EUGENE modules for the Punjabi language have been documented. This chapter highlights the framework of EUGENE tool which is used for NLization. The working of the developed NLization module using EUGENE has been presented with example sentences.

This chapter highlights the use of NLization/EUGENE module from the developed question answering system perspective and gives details about how generation module of the proposed question answering system can be used to invoke NLization/EUGENE module of target natural language.

After explaining about EUGENE, generation and NLization/EUGENE modules, the state of art of NLization/EUGENE module of Punjabi language has been discussed in this chapter. The achievements and contributions of the developed NLization/EUGENE module for the Punjabi language have been highlighted in this chapter.

Chapter 7

Results and Discussion of Proposed QA System

As discussed in previous chapters, on the proposed QA system, the question asked by the user is converted to UNL and the developed QA system crawls the UNL of question and UNL of corpus to find the answer in target natural language. This chapter discusses about the evaluation of the proposed QA system. It discusses about the corpus, metrics, empirical results, and error analysis of the proposed QA system and its comparison with other QA systems. For testing the proposed QA system, the questions were framed on UNL-EOLSS corpus, and Agro-Explorer corpus.

7.1 Corpus Details

The UNL-EOLSS corpus for experimentation has been taken from World's largest online publication repository "The Encyclopedia of Life Support Systems (EOLSS)" that is available at *http://www.eolss.net*. This repository consists of articles related to the maintenance, health, and future of the web of life on planet earth. The UNDL foundation, under the project UNL-EOLSS created the content of 30 articles of the Encyclopedia of Water, one of the many encyclopedias of EOLSS[1]. The UNDL foundation has made available UNL-EOLSS corpus which has 25 articles and 12,917 sentences[2]. To test the proposed QA system, questions were framed on these 25 articles.

Agro-Explorer is an agricultural domain corpus developed by CFILT, IIT Bombay[3] and has been spitted into 7 sections. It comprises of total 240 complex sentences. Questions were framed on these 7 sections so as to test the proposed QA system on another corpus.

Harabagiu and Moldovan (2003) had identified the following six QA question types, *i.e.*, Factoid, List, Defintion, How, Why, and Polar (Yes-No) [59].

Factoid questions are like *"Who discovered the oxygen?", or "When did Hawaii become a state?".*

---

[1]http://www.unlweb.net/wiki/EOLSS

[2]http://www.undlfoundation.org/eolss/corpus/UNL.txt

[3]http://www.cfilt.iitb.ac.in

List questions are like *"Which countries export oil?" or "What are the regions preferred by the Americans for holidays?".*

Definition questions are like *"What is a quasar?" or "What is a question-answering system?".*

How questions are like *"How seawater intrusion can be prevented?".*

Why questions are like *"Why regeneration is more thorough?".*

Polar questions are like *"Is Hydroponics a method of soilless gardening?".*

In order to test the proposed QA system, all the 400 questions which were framed on Agro-Explorer and UNL-EOLSS corpus covers all the major part of speeches and all QA question types discussed above.

7.2 Metrics Used for Evaluation

Hirschman and Gaizauskas (2001) had suggested the following important metrics for the evaluation of a QA system [65].

- **Conciseness:** It means that the answer should not consist of irrelevant/extraneous information.
- **Relevance**: It specifies that the answer should be a response to the question.
- **Correctness:** It specifies that the answer should be factually correct.

Apart from these metrics, **Precision** and **Recall** have also been long used for evaluating QA systems [52][23]. Precision is the measure of accuracy whereas Recall is the measure of its exhaustivity. Precision means what percentage of answers given by the QA system is relevant and is calculated by the formula given in (7.1). Recall means what percentage of relevant answers is returned by the QA system and is calculated by the formula given in (7.2).

$$Pr\,e\,cision \quad = \quad \frac{Number \quad of \quad relevant \quad retrieved}{Total \quad retrieved}$$

$$Recall = \frac{Number\ of\ relevant\ retrieved}{Total\ relevant\ items}$$

F-Measure or F1-Score is the harmonic mean of Precision and Recall and is calculated by the formula given in (7.3).

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad ...(7.3)$$

7.2.1 Methodology to Calculate these Metrics Values

The questions that were framed out of the two corpora were assumed to have only one correct answer which can be present in any sentence of these corpora.

Consider a scenario, in which user asks a question to the QA system and the QA system responded with 3 answers out of which only one answer is right.

In this case the **Relevance** of this question would be 1/3 = 33.3% because 1 out of 3 answers is relevant. The **Precision** of this will be same as Relevance as out of 3 answers 1 answer is relevant. This is true for every case. The Relevance/Precision of the complete QA system would be the weighted average of Relevance/Precision of all the 400 questions asked with weights equal to the total number of questions per question type as given in Table 7.1.

Conciseness is calculated by the formula given in (7.4).

$$Conciseness = \frac{Total\ concise\ answers\ obtained}{Total\ questions\ asked} \times 100 \quad ...(7.4)$$

As described earlier, an answer is said to be concise if it does not have any irrelevant or extraneous information. An answer which is not concise will have Relevancy/ Precision equal to 0.

Correctness is calculated by the formula given in (7.5).

$$Correctness = \frac{Total\ factually\ correct\ answers\ obtained}{Total\ questions\ asked} \times 100 \quad ...(7.5)$$

Since all the answers to the questions had been extracted from the UNL repository (which is assumed to have correct UNL) and no manipulation is done by the developed QA system, therefore all were factually correct and hence ‘Correctness’ is 100%.

Since, the proposed system works on UNL of input question and UNL of corpus, therefore the **Recall** of the proposed system is 1 because, by design, UNL crawler will definitely find an answer if it is present in the corpus. However, if the UNL of question or corpus is wrong, then it will reduce the Recall.

Since for the polar questions, answers contain only 'Yes' or 'No', therefore Conciseness and Relevance/Precision is not applicable to this and is evaluated as 1 always.

7.3 Empirical Results

For testing the proposed QA system, 400 questions were framed and asked through the developed interface as illustrated in Figure 4.32 of Chapter 4 of this thesis. All the sentences in the UNL-EOLSS corpus, and Agro-Explorer corpus were converted to UNL and saved as UNLEolss.txt, and AgroExplorer.txt files respectively (which form the UNL repository) as a part of the developed solution. Out of these 400 questions, total number of missing type questions (questions in which a node in the UNL graph has an argument '00.@wh' as described in section 5.1 of Chapter 5 of this thesis) were 300, Polar (Yes-No type) questions were 70 and non-missing type questions (questions in which a node in the UNL graph has an argument 'xyz@wh' as described in section 5.1 of Chapter 5 of this thesis) were 30. Table 7.1 presents the count of different types of questions on which the proposed system has been tested.

Table 7.1: Types of Questions

Sr No.	**Question Type**	**Count**	**Category**
1.	Polar (Yes-No)	70	Polar question
2	What	60	Missing type question
3.	Who	50	
4.	Where	40	
5.	When	40	
6.	Why	40	
7.	Which	40	
8.	How	30	
9.	What	30	Non-Missing type question
Total	-	**400**	

After testing the proposed QA system on 400 questions, it has been observed that its 'Conciseness' is 89.5% as out of 400 questions, answers to 42 questions had extra information. The value of Relevance/Precision of these 42 questions would be 0.

Apart from these 42 questions, there were 17 more questions, whose responses were in the form of multiple answers. Table 7.2 shows the question wise values of Conciseness, Relevance/Precision, Recall and F-Measure of the developed QA system and has been calculated with the help of formulae given in 7.1, 7.2, 7.3, and 7.4.

Table 7.2: Conciseness, Relevance/Precision, Recall and F-Measure of the Developed QA System

Sr No.	**Question Type**	**Count**	**Conciseness**	**Relevance/ Precision**	**Recall**	**F-Measure**
1.	What	90	0.84	0.81	1	0.895
2.	Polar (Yes-No)	70	1.0	1.0	1	1
3.	Who	50	0.86	0.82	1	0.901
4.	Where	40	0.90	0.86	1	0.924
5.	When	40	0.95	0.93	1	0.963
6.	Why	40	0.87	0.85	1	0.918
7.	Which	40	0.85	0.80	1	0.888
8.	How	30	0.86	0.81	1	0.895
Total	**-**	**400**	**0.895**	**0.864**	**1**	**0.927**

Figure 7.1 shows the values of Conciseness, Correctness, Relevance/Precision, Recall, and F-Measure of the developed QA system.

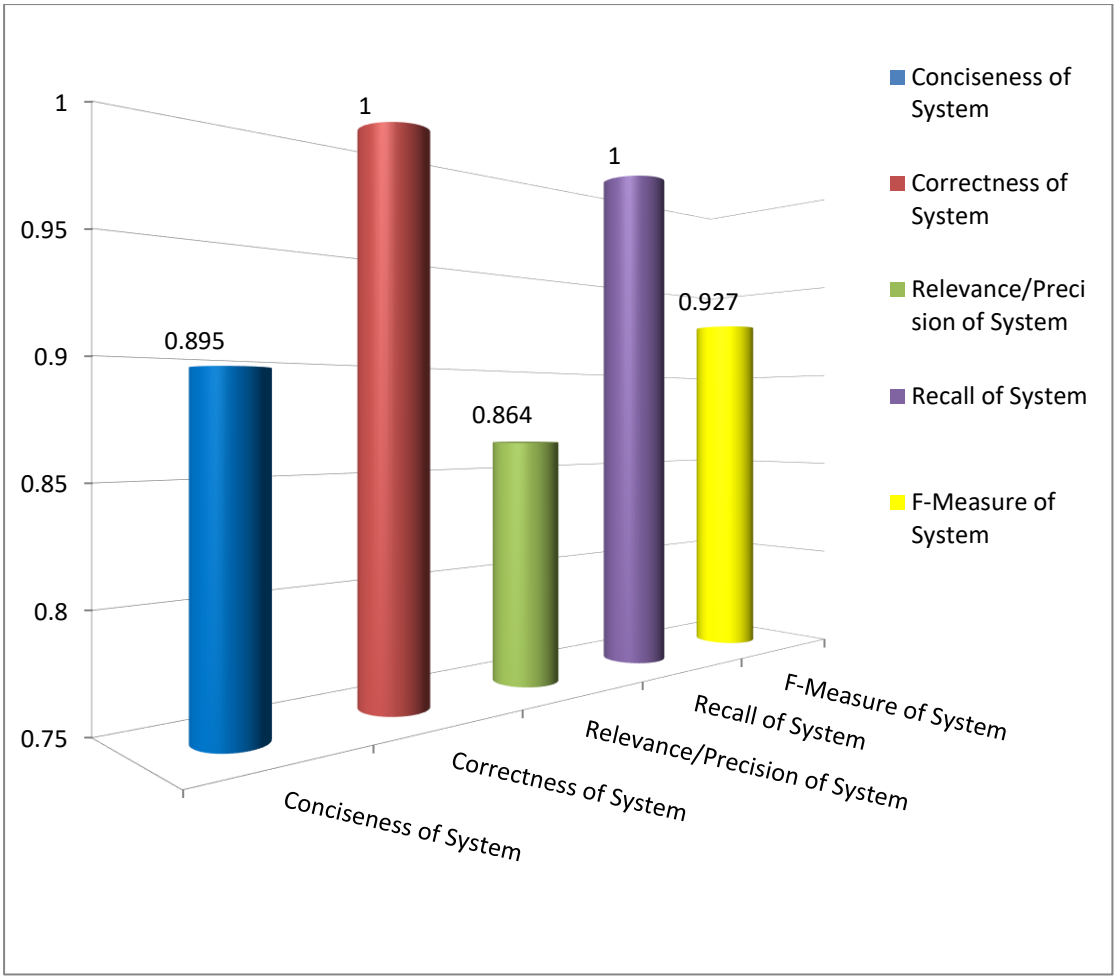


Figure 7.1: Measures of Conciseness, Correctness, Relevance/Precision, Recall, and F-Measure

As described earlier in section 7.2.1, since the proposed system works on UNL of input and UNL of corpus, therefore the Recall of the proposed system is 1 because, by design, UNL crawler will definitely find an answer if it is present in the corpus.

Since for the polar questions, answers contain only 'Yes' or 'No', therefore Conciseness and Relevance/Precision is not applicable to this and is evaluated as 1 always.

7.4 Comparison of Results with other QA Systems

As discussed in literature review of this thesis, unlike QA systems proposed using non UNL techniques, there are very few UNL based QA/search engines/information retrieval systems that have been proposed by various researchers. Although accuracy of some of them had been given, yet no information regarding the calculation of those results, testing methodology, and corpus details *etc.* had been reported. Therefore, the

F-Measure of our system has been compared with accuracies of other important proposed QA systems as well.

However, for UNL based QA system developed by Cardenosa *et al.* (2009) [33] information regarding accuracy, number of questions and corpus has been given. The comparison of their total number of questions (on which testing has been performed) of each type has been done with our proposed system and is shown in Figure 7.2.

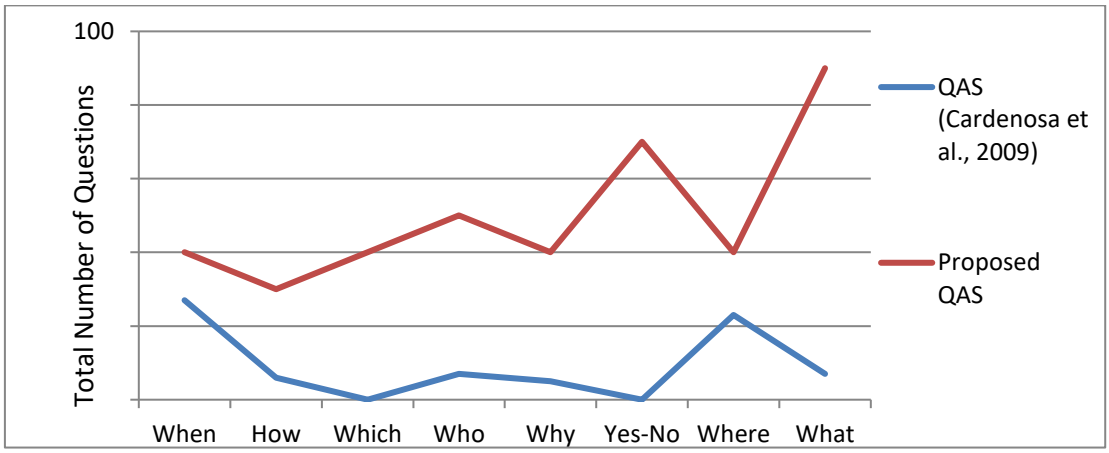


Figure 7.2: Comparison of the Total Number of Questions Used for Testing Different UNL Based QAS

As shown in Figure 7.2, the total number of questions used by Cardenosa *et al.* (2009) [33] for testing their system is significantly less. Apart from the accuracy of QA system developed by Cardenosa *et al.* (2009) [33], accuracies of UNL based QA systems developed by Goel (2016) [54], and Shukla *et al.* (2004) [134] had also been reported. The F-Measure of the developed QAS as compared with the accuracy of UNL based QAS developed by Cardenosa *et al.* (2009) [33], Goel (2016) [54], and Shukla *et al.* (2004) [134] is shown in Figure 7.3. The accuracies of system developed by Cardenosa *et al.* (2009) [33], Goel (2016) [54], Shukla *et al.* (2004) [134], and F-Measure of our system is 0.826, 0.82, 0.6, and 0.927 respectively.

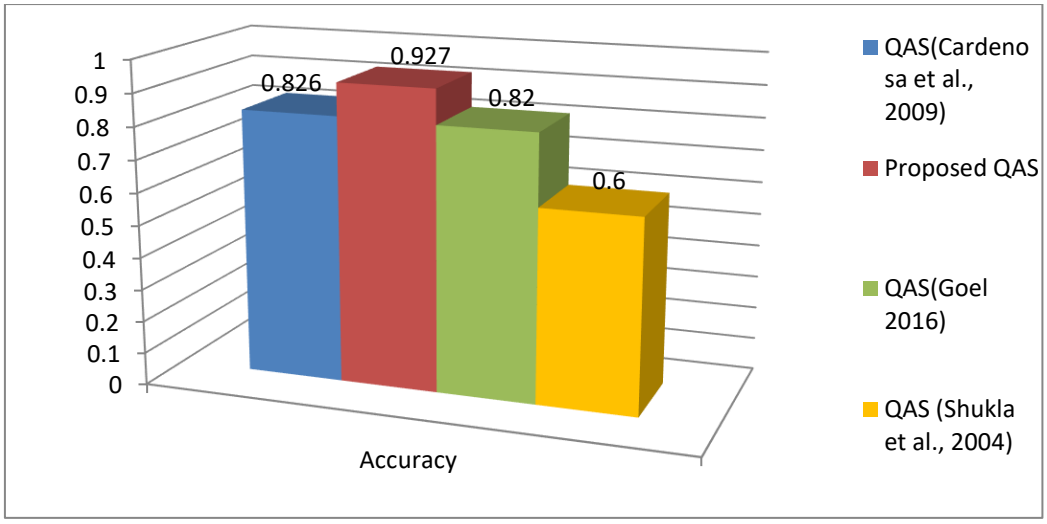


Figure 7.3: Comparison of the Accuracy of Different UNL Based QAS

Since accuracy of only three UNL based QA systems has been reported, so in order to evaluate and compare proposed QA system, its F-Measure has also been compared with other important and high accurate QA systems which are developed using different techniques as shown in Figure 7.4.

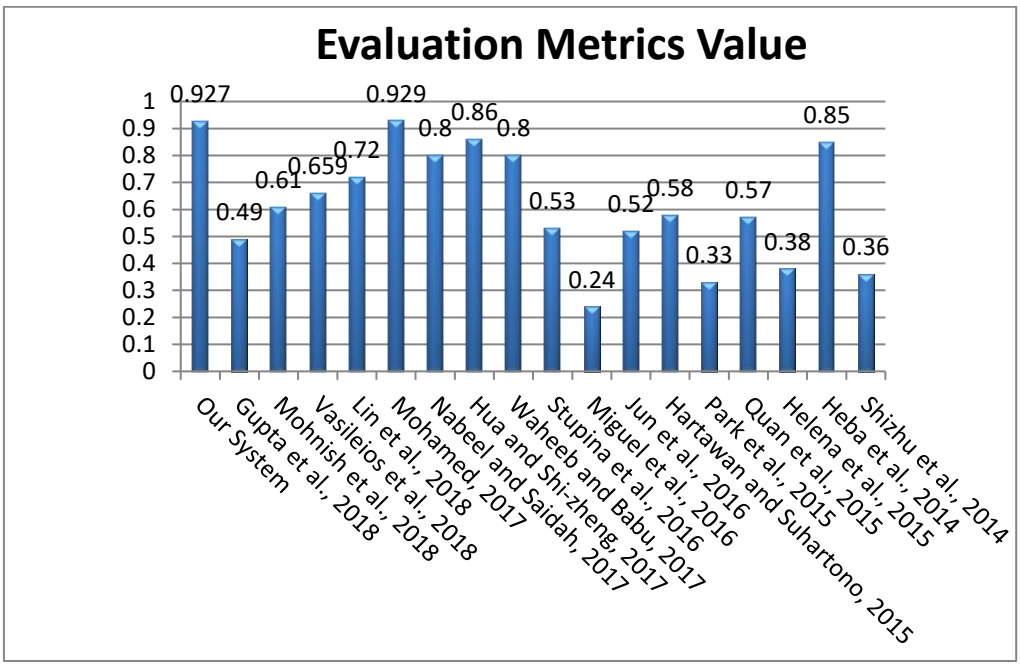


Figure 7.4: Comparison of Evaluation Metrics Value of Our System with Different Developed QAS

After doing an exhaustive survey on the available QA systems and comparing the accuracy/features of our proposed QA system with other QA systems, it has been observed that our proposed QA system is having the second best accuracy (after the system proposed by Mohamed 2017 [101]) and the proposed system will have a major

impact in research and development of QA systems. The proposed system is web based and its working component's details have been made available and it can integrate other UNL based applications, and supports multilinguism (without changing the architecture and code). This is a major breakthrough in development of NLP applications.

7.5 Error Analysis

Error analysis plays a vital role in analyzing the limitations of an application and proposed research. It gives an understanding and provides an oppourtunity to improve the accuracy of the system. The following sub-sections present the detailed explaination about the error analysis of the proposed QA system.

7.5.1 Relevance

As mentioned earlier, Relevance specifies that the answer should be a response to the question. The proposed system has achieved 'Relevance' of 0.864 when tested on 400 questions covering each question type like What, Polar, Who, Where, When, Why, Which, and How. To understand the situation that caused reduction in Relevance, let us consider that the user asks the following question as given in (7.6).

What is agriculture? …(7.6)

Cleaned UNL of this question after preprocessing is given in (7.7).

{unl}
aoj(00.@wh;agriculture)
{/unl} …(7.7)

In order to provide the answer to this question, as discussed in Chapter 5 of this thesis, the QA system will try to look for '*aoj*' relation whose second argument is '*agriculture'.*

Now, Agro-Explorer corpus contains cleaned UNL (after preprocessing) as depicted in Figures 7.5, 7.6, and 7.7 for the 3 sentences as given in (7.8), (7.9), and (7.10) respectively.

"Agriculture is art, science, and industry of managing the growth of plants and animals for human use." …(7.8)

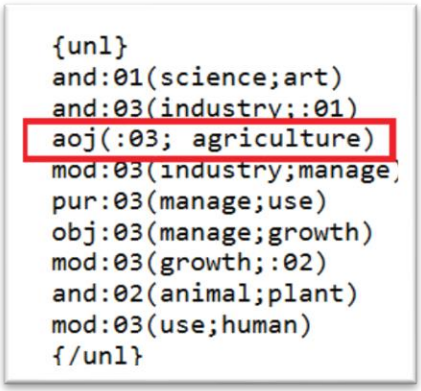


Figure 7.5: UNL of Example Sentence (7.8)

"In a broad sense, agriculture includes cultivation of the soil and growing and harvesting crops and breeding and raising livestock and dairying and forestry."

...(7.9)

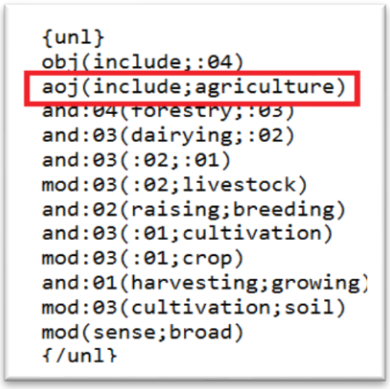


Figure 7.6: UNL of Example Sentence (7.9)

"Modern agriculture depends heavily on engineering and technology and on the biological and physical sciences." ...(7.10)

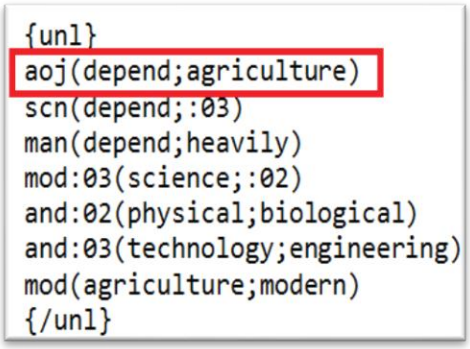


Figure 7.7: UNL of Example Sentence (7.10)

Now, since UNL repository has 3 UNL sentences (as shown in Figures 7.5, 7.6 and

7.7) with '*aoj*' relation, and '*agriculture*' as the second argument, therefore, we get three answers of the question asked in example sentence (7.6). The answer given by the developed QA system is as shown in Figure 7.8. Due to this, the Relevance/Precision of this question would be 1/3 = 0.33 (refer equation 7.1). Therefore, the proposed QA system is not 100% relevant and its Relevance/Precision comes out to be 0.864 when tested on 400 questions.

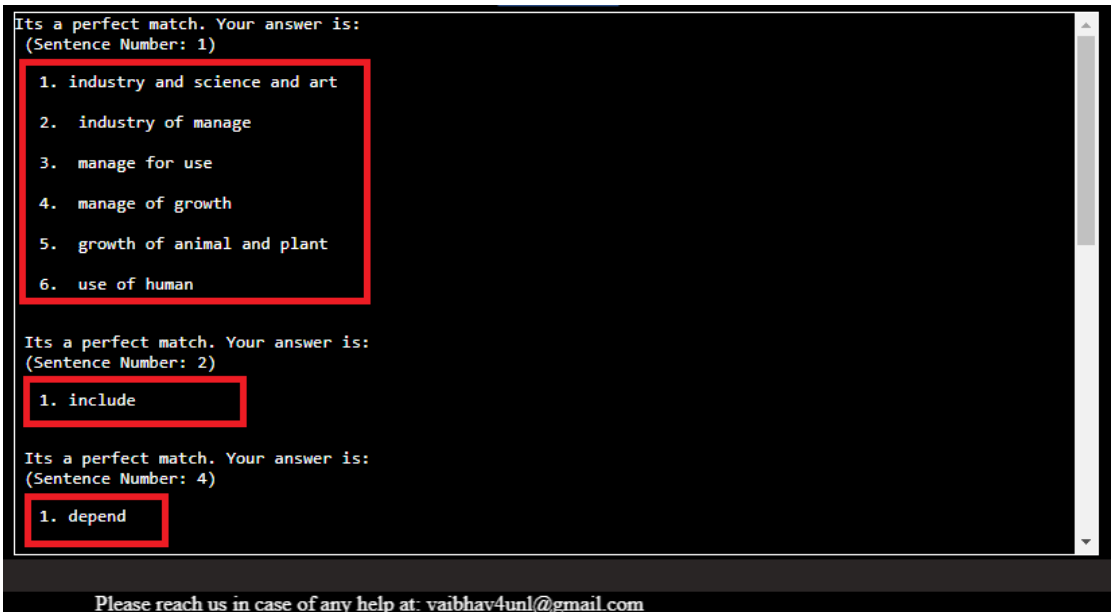


Figure 7.8: Answer of Question Asked in Example Sentence (7.6)

7.5.2 Conciseness

As mentioned earlier, Conciseness specifies that the answer should not contain irrelevant or extraneous information. The proposed system has achieved 'Conciseness' of 0.895 when tested on 400 questions. To understand the situation that caused reduction in Conciseness, let us consider that the user asks the following question as given in (7.11).

What genetics has made? …(7.11)

Cleaned UNL of this question after preprocessing is given in (7.12).

{unl}
aoj(make,genetics.@wh)
{/unl} …(7.12)

Now, as described in Chapter 5 of this thesis, QA system tries to answer this question by looking all the relations whose first argument is '*make'*.

Agro-Explorer corpus contains cleaned UNL (after preprocessing) as shown in Figure

7.9 for the sentence given in (7.13).

"Genetics has also made a science of livestock breeding." …(7.13)

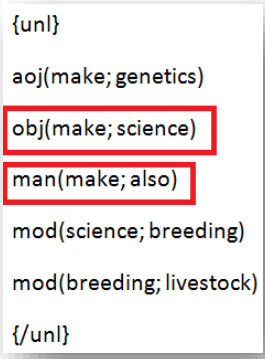


Figure 7.9: UNL of Example Sentence (7.13)

Now, since UNL repository has this UNL sentence (as shown in Figure 7.9) with 'make' as the first argument, therefore, we get two answers of the question asked in example sentence (7.11). The answer given by the developed QA system is as shown in Figure 7.10. Due to this, this answer is not concise and its Conciseness is 0. Therefore, the Conciseness of the proposed QA system is not 100% and its Conciseness comes out to be 0.895.

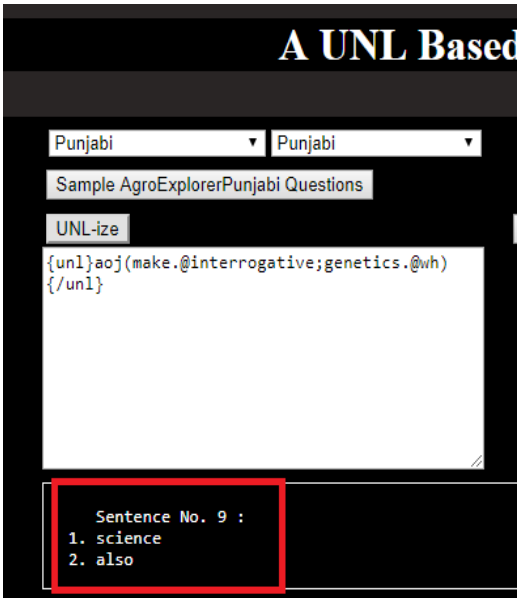


Figure 7.10: Answer of Question Asked in Example Sentence (7.11)

The system has been developed to support multi UNL document as an answer file. Therefore, UNL sentence in which the answer is present will come at the top (based on ranking given by the optimizer, refer to section 5.2 of Chapter 5 of this thesis).

Table 7.3 shows some of the example questions and answers given by the developed QA system.

Table 7.3: Example Questions Asked to the Proposed QA System

Sr No.	Natural Language Sentence	Corpus Name	Question Asked	QA System (Optimizer) Output
1.	Natural water resources represent a vital biosphere component HTM1 an economic and industrial resource that possesses indispensable consumer properties.	UNL-EOLSS	Natural water resources represent what	It's a perfect match. Your answer is: (Sentence Number: 1) 1. Component
2.			Natural water resource represent which component	It's a perfect match. Your answer is: (Sentence Number: 1) 1. Biosphere
3.			Is biosphere component vital	Yes
4.	Natural water resources represent a vital biosphere component HTM1 an economic and industrial resource that possesses indispensable consumer properties.	UNL-EOLSS	Natural water resources represent a vital biosphere component	Sorry, we cannot find any question. Please ask the question again.
5.	Modern agriculture depends heavily on engineering and technology and on the biological and physical sciences.	Agro-Explorer	Agriculture depends on what	It's a perfect match. Your answer is: (Sentence Number: 1) 1. science of physical and biological 2. technology and engineering 3. heavily
6.	Agriculture is art, science, and industry of managing the growth of plants and	Agro-Explorer	Is agriculture an art	Yes

	animals for human use.			
7.	In a broad sense, agriculture includes cultivation of the soil and growing and harvesting crops and breeding and raising livestock and dairying and forestry.	Agro-Explorer	What does agriculture include	(Sentence Number: 1) 1. forestry and dairying and raising and breeding 2. raising and breeding and harvesting and growing 3. raising and breeding of livestock 4. harvesting and growing and cultivation 5. harvesting and growing of crop 6. cultivation of soil
8.	Each of irrigation and drainage and conservation and sanitary engineering is important in successful farming.	Agro-Explorer	What is important in successful farming	It's a perfect match. Your answer is: (Sentence Number: 1) 1. engineering and conservation and drainage and irrigation 2. engineering of sanitary
9.	Agricultural chemistry deals with other vital farming concerns, such as the application of fertilizer and insecticides and fungicides and soil makeup and analysis of agricultural products and nutritional needs of farm animals.	Agro-Explorer	Who deals with vital farming concerns	It's a perfect match. Your answer is: (Sentence Number: 1) 1. chemistry
10.			Which chemistry deals with vital farming concerns	It's a perfect match. Your answer is: (Sentence Number: 1) 1. agricultural
11.	Plant breeding and genetics contribute immeasurably to farm productivity.	Agro-Explorer	What contributes to farm productivity	It's a perfect match. Your answer is: (Sentence Number: 1) 1. genetics and breeding 2. breeding of plant
12.	Genetics has also made a science of livestock breeding.		What genetics has made	(Sentence Number: 1) 1. science 2. also

		Agro-Explorer		
13.			genetics has made science of what	1. breeding
14.			genetics has made science of which breeding	1. livestock
15.	Water consumption on Earth during the twentieth century increased sevenfold while the population only increased three times.	UNL-EOLSS	Where water consumption has increased	1. earth
16.			In which century water consumption on earth has increased	It's a perfect match. Your answer is: (Sentence Number: 1) 1. twenty
17.	Water quality is the most important factor of ecological safety and stability in agriculture development.	UNL-EOLSS	Water quality is the most important factor of what	It's a perfect match. Your answer is: (Sentence Number: 1) 1. stability and safety
18.			Water quality is the most important factor of stability and safety of what	It's a perfect match. Your answer is: (Sentence Number: 1) 1. ecological

The results of the proposed system are very motivating. Such a UNL based (online available) QA system which can support foreign languages will definitely be a major step in NLP for removing the language barriers.

Chapter Summary

In this chapter, the detailed information of experimentation and evaluation of the proposed QAS has been documented. This chapter gives the details of corpora used for testing the developed question answering system. This chapter introduces the evaluation metrics *viz*. Conciseness, Relevance, Correctness, Precision, Recall, and F-Measure for evaluating the developed QAS. The details regarding the total number of questions and their types have been highlighted in this chapter. The Conciseness, Relevance, Correctness, Precision, Recall, and F-Measure of the developed QAS came out to be 89.5%, 86.4%, 100%, 86.4%, 100%, and 92.7% respectively.

The analysis/comparison of the developed QAS on the basis of F-Measure/accuracy has been done with different question answering systems. The error analysis of Relevance and Conciseness has also been performed and explained in this chapter. The testing methodology and example questions along with their answers have also been listed in this chapter.

Chapter 8

Conclusion and Future Scope

8.1 Conclusion

As a part of this thesis, various gaps were identified in the existing question answering systems and objectives were framed to address these identified gaps. As discussed in Chapter 2 (Background Theory and Literature Review) of this thesis, analysis of various QA systems has been done based on the following parameters:

- Corpus used for testing
- Evaluation metric
- Evaluation metrics value
- Domain
- Is working algorithms of all components explained
- Is question answering system available online
- Does QAS support a multiple language
- Can QAS be extended to support other foreign languages
- Can question answering system integrate other NLP applications
- Is source code available for the given question answering system

A framework for UNL based question answering system has been proposed to meet all the objectives. Based on this framework, the proposed question answering system has been developed and tested for Punjabi language. The Punjabi language resources for UNLization and NLization modules have been created. The proposed question answering system meets all the objectives as discussed in section 1.9 of Chapter 1.

A public platform for developing UNL based language-independent applications has been developed and tested. The analysis and generation modules of this platform invoke IAN and EUGENE module of the source and target natural language (Punjabi in this case) for UNLization and NLization respectively. This web-based platform has been developed using HTML, CSS, jQuery, and C#. The proposed QA system components, *i.e.*, UNL repository, UNL crawler, optimizer, and the ranking mechanism have been developed and integrated with this platform. IAN/UNLization and EUGENE/NLization modules have been developed for Punjabi natural language. UNDL foundation had conducted a series of Olympiads over the years. These Olympiads for UNLization and NLization were held so as to find the best IAN and EUGENE modules for the participating languages across the globe and check their respective accuracies. Till date, more than 33 languages are part of this UNL

programme. The UNLization module for Punjabi language had been submitted for UNL Olympiad II, III, and IV conducted by UNDL foundation in July 2013, March 2014, and November 2014 for UC-A1, UGO-A1, and AESOP-A1 corpora respectively [129][150][49]. The UC-A1, UGO-A1, and AESOP-A1 corpora were provided by UNDL foundation which covered all the major part-of-speech.

The language selected for the proposed question answering system, *i.e*., Punjabi, had been selected in top 10 (Based upon the F-Measures) UNLization grammars for Olympiad II, while it was selected in top 5 best grammars for Olympiad III and Olympiad IV. The F-Measure of Punjabi IAN module was 0.97, 0.99, and 1.00 for Olympiad II, Olympiad III, and Olympiad IV respectively.

Punjabi language has been selected in top 5 (based upon the F-Measures) NLization grammars for Olympiad III, and Olympiad IV. The F-Measure of Punjabi language EUGENE module was 0.996 and 1.00 for Olympiad III and Olympiad IV respectively.

The evaluation metrics *viz*. Conciseness, Relevance, Correctness, Precision, Recall, and F-Measure has been used for evaluating the developed question answering system. For testing the proposed QA system, the questions were framed on UNL-EOLSS corpus, and Agro-Explorer corpus.

The UNL-EOLSS corpus for experimentation has been taken from World's largest online publication repository "The Encyclopedia of Life Support Systems (EOLSS)" that is available at *http://www.eolss.net*. This repository consists of the articles related to the maintenance, health and future of the web of life on planet earth. The UNDL foundation, under the project UNL-EOLSS created the content of 30 articles of the Encyclopedia of water, one of the many encyclopedias of EOLSS. The UNDL foundation has made available UNL-EOLSS corpus which has 25 articles and 12,917 sentences. To test the proposed QA system, questions were framed on these 25 articles.

Agro-Explorer is an agricultural domain corpus developed by CFILT, IIT Bombay and has been spitted into 7 sections. It comprises of total 240 complex sentences. The questions were framed on these 7 sections so as to test the proposed QA system on another corpus.

For testing the proposed QA system, 400 questions were framed and asked through the developed interface. The Conciseness, Relevance, Correctness, Precision, Recall,

and F-Measure of the developed question answering system came out to be 89.5%, 86.4%, 100%, 86.4%, 100%, and 92.7% respectively.

The major outcomes of this research work are as follows.

i. Study and comparison of existing QASs has been done on the basis of important parameters like corpus, evaluation metrics used, evaluation metrics values, and domain *etc.*

ii. Various gaps in the existing QASs has been identified on the basis of important parameters like, availability of source code, scalability, online availability, multilingual support, is working algorithms of all components explained, and can question answering system integrate other NLP applications *etc.*

iii. A web based language independent architecture for QAS has been proposed to address the identified gaps.

iv. A web based interface for the QAS has been developed on top of the proposed architecture. UNL crawler and optimizer has been developed and integrated in this interface.

v. UNLization/IAN module for Punjabi language has been developed.

vi. NLization/EUGENE module for Punjabi language has been developed.

vii. Analysis and generation dictionaries for Punjabi language have been created.

viii. The developed QAS has social significance. It has been tested on UNL-EOLSS and Agro-Explorer corpora which are based on water resources and farming.

ix. Developed 'Share' functionality by which worldwide users can share their UNLization/IAN and NLization/EUGENE resources so as to make their natural language available for use (in the developed QAS).

x. Implemented OTP based authentication feature for 'Share' functionality.

8.2 Limitations of the Developed QA System

The framework of the proposed QA system is so much flexible that any natural language which is or will be the part of UNL programme will be supported by the proposed architecture. The developed QA system's code base need not be changed in order to incorporate those languages. However, the developed system has the following limitations:

- It is assumed that corpus's UNL will have numbers in the form of words and not figures. For example, 19 will be present as 'nineteen' in the UNL corpus.
- The developed system is dependent on API's (for calling IAN and EUGENE resources) developed by UNDL foundation.
- The 'Conciseness', 'Relevance', 'Precision', and 'F-Measure' metric values changes with the number and type of questions for any question answering system.

8.3 Future Scope

The 'Conciseness', 'Relevance', 'Precision', and 'F-Measure' metric values of the proposed QA system can be improved. IAN and EUGENE modules can be enriched for Punjabi natural language so that their F-Measure can be increased. Since a public platform for developing language-independent applications has been developed and tested, therefore other NLP applications which are UNL based like sentiment analysis, text summarization, machine translation *etc.* can be developed and integrated with this. The developed system can be extended to support the feature to upload the UNL corpus by the user so that questions can be asked by the worldwide audience.

Certifications in UNL

Certificate of Language Engineering CLEA250

UNDL FOUNDATION

Universal Networking Language

CLEA250

Certificate of Language Engineering Aptitude

The UNDL Foundation certifies that

vaibhav agarwal

has successfully completed the exercises leading to the Certificate of Language Engineering Aptitude (CLEA250), offered online through VALERIE, the Virtual Learning Environment for UNL.

Geneva, June 24, 2013

Tarcisio G. Della Senta, PhD
President
UNDL Foundation

Ronaldo T. Martins, PhD
Language Resources Manager
UNDL Foundation

Cleared CUP 500 Certificate of Proficiency in UNL.

CUP 500

Certificate of Proficiency in UNL

The UNDL Foundation certifies that

vaibhav agarwal

has successfully completed the exercises leading to the Certificate of Proficiency in UNL (CUP500), offered online through VALERIE, the Virtual Learning Environment for UNL.

Geneva, July 15, 2012

Tarcisio G. Della Senta, PhD
President
UNDL Foundation

Ronaldo T. Martins, PhD
Language Resources Manager
UNDL Foundation

Publications

Research Papers Published

- V. Agarwal, P. Kumar. “UNLization of Punjabi Text for Natural Language Processing Applications.” Springer: Sadhana, Academy Proceedings in Engineering Sciences, DOI: https://doi.org/10.1007/s12046-018-0824-z, May, 2018, IF: 0.592.

- V. Agarwal, P. Kumar. “A public platform for developing language-independent applications.” Digital Scholarship in the Humanities (previously known as Literary and Linguistic Computing), Oxford University Press, DOI: https://doi.org/10.1093/llc/fqwx063, February, 2017, IF: 0.563.

- V. Agarwal, P. Kumar. “A Multilingual Cross-Domain Client Application Prototype for UNLization and NLization for NLP Applications.” Digital Scholarship in the Humanities (previously known as Literary and Linguistic Computing), Oxford University Press, DOI: https://dx.doi.org/10.1093/llc/fqw022, April 2016, IF: 0.563.

Research Papers Under Revision

- V. Agarwal, P. Kumar. “A Framework for Web Based Language Independent Semantic Question Answering System.” Semantic Web Journal, IOS Press.

References

[1] Abderrazzak, S., Qbadou, M., Youssfi, M., & Akef, F. (2018). A syntactic and semantic multi-agent based question answering system for collaborative e-learning. In *Optimization and Applications (ICOA), 2018 4th International Conference on* (pp. 1-4). IEEE.

[2] Abdullah, M.M., & Abdel-Kader, R. F. (2011). Qasyo: A question answering system for yago ontology. *International Journal of Database Theory and Application*, *4*(2), 99-112.

[3] Adly, N., & Ansary, S. (2009). Evaluation of Arabic machine translation system based on the Universal Networking Language. *Natural Language Processing and Information Systems, Lecture Notes in Computer Science*, 5723, 243-257.

[4] Agarwal, V., & Kumar, P. (2018). UNLization of Punjabi text for natural language processing applications. *Sādhanā*, 43-87.

[5] Agarwal, V., & Kumar, P. (2017). A public platform for developing language-independent applications. *Digital Scholarship in the Humanities*, 33(1), 1-5.

[6] Agarwal, V., & Kumar, P. (2016). A Multilingual Cross-Domain Client Application Prototype for UNL-ization and NL-ization for NLP Applications. *Digital Scholarship in the Humanities*, 32(3), 461-475.

[7] Agarwal, V., & Kumar, P. (2013). UNLization of Numbers and Ordinals in Punjabi with IAN. *International Journal on Natural Language Computing (IJNLC)*, 2(3),43-48.

[8] Agarwal, V., & Kumar, P. (2013). UNLization of Punjabi with IAN. *In Proc. of the 11th workshop on Asian Language Resources*, Nagoya, Japan, 32-39.

[9] Agarwal, V. (2013). *UNLization of Punjabi with IAN* (M.E dissertation), Thapar University, Patiala.

[10] Ansary, S., Nagi, M., & Adly, N. (2006). Towards a Language Independent Universal Digital Library. In *The Second International Conference on Universal Digital Libraries (ICUDL 2006). Alexandria, Egypt 17-19 November 2006*.

[11] Allam, A.M.N., & Haggag, M.H. (2012). The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS), 2*(3).

[12] Ali, M.N.Y., Sorwar, G., & Shamsujjoha, M. (2015). Formation of Word Dictionary of Bangla Vowel Ended Roots for First Person for Universal Networking Language. *International Conference on Information and Knowledge Engineering*, Las Vegas, USA.

[13] Amit, M., & Sanjay, K.J. (2016). A survey on question answering systems with classification. *Journal of King Saud University-Computer and Information Sciences, 28*(3), 345-361.

[14] Amrita, S., Vardaan, P., Mitesh, M. K., Karthik, S., & Sarath, C. (2018). Complex Sequential Question Answering: Towards Learning to Converse Over Linked Question Answer Pairs with a Knowledge Graph. *arXiv preprint arXiv:1801.10314*.

[15] Arvind,K.T., & Rajeev, S. (2014). A Survey of Computational Intelligence Techniques in Protein Function Prediction. *International Journal of Proteomics*, 845479, 1-22.

[16] Ask Jeeves. Wikipedia.[Online]. Available: http://en.wikipedia.org/wiki/Ask.com

[17] Avetisyan, A., & Avetisyan, V. (2010). LOOK4: Enhancement of web search results with Universal Words and WordNet. In *Proc. 5th Int. Conf. on Global WordNet* (pp. 1-5).

[18] Baudiš, P. (2015). YodaQA: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering* (pp. 1156-1165).

[19] Bekios, J., Boguslavsky, I., Cardeñosa, J., & Gallardo, C. (2007). Using wordnet for building an interlingua dictionary. In *Fifth International Conference INFORMATION RESEARCH AND APPLICATIONS* (p. 39).

[20] Ben, H., Peter, C., & Hannaneh, H. (2015). Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 851-861).

[21] Bértoli Jr, L., Da Luz, R. P., & Bastos, R. C. (2005). A web platform using UNL: CELTA's showcase. *Universal Networking Language: Advances in Theory and Applications*, 276.

[22] Bhattacharyya, P. (2001). Multilingual information processing through Universal Networking Language. In *Indo UK Workshop on Language Engineering for South Asian Languages* (pp. 1-10).

[23] Bilotti, M. W., & Nyberg, E. (2006). Evaluation for scenario question answering systems. In *Proceedings of the International Conference on Language Resources and Evaluation*.

[24] Blanc, E. (2005). About and around the French Enconverter and the French Deconverter. *Universal Network Language: Advances in Theory and Applications*, *12*, 157-166.

[25] Boguslavsky, I., Iomdin, L., & Sizov, V. (2005). Interactive enconversion by means of the ETAP-3 system. In *Proceedings of the International Conference on the Convergence of Knowledge, Culture, Language and Information Technologies", Alexandria*.

[26] Boitet, C., Bhattacharyya, P., Blanc, E., Meena, S., Boudhh, S., Fafiotte, G., & Vacchani, V. (2007). Building Hindi-French-English-UNL resources for SurviTra-CIFLI, a linguistic survival system under construction. In *Seventh international symposium on natural language processing* (p.7).

[27] Boudhh, S., & Bhattacharyya, P. (2009). Unification of universal word dictionaries using WordNet ontology and similarity measures. In *proceedings of the 7th International Conference on Computer Science and Information Technologies*.

[28] Bronnenberg, W. J. H. J., Bunt, H. C., Landsbergen, S. J., Scha, R. J., Schoenmakers, W. J., & Van Utteren, E. P. C. (1980). The question answering system PHLIQA1. *Natural Language Question Answering Systems, Natural Communication with Computers*.

[29] Bueno, T. C., Hoeschl, H. C., Bortolon, A., Mattos, E. S., Santos, C., & Barcia, R. M. (2005). Knowledge engineering suite: a tool to create ontologies for automatic knowledge representation in intelligent systems. *Universal Networking Language: Advances in Theory and Applications,* 337.

[30] Calculation of F-Measure [Online]. Available: http://www.unlweb.net/unlarium/index.php?tools=fmeasure

[31] Cameron, R. D. (1998). Rex: Xml shallow parsing with regular expressions. *Markup Languages*, *1*(3), 61-88.

[32] Cardeñosa, J., Gallardo, C., & Iraola, L. (2005). An XML-UNL Model for Knowledge-Based Annotation. *Research on Computing Science*, *12*, 300-308.

[33] Cardeñosa, J., Gallardo, C., & Miguel, A. (2009). Interlingual information extraction as a solution for multilingual QA systems. In *International Conference on Flexible Query Answering Systems* (pp. 500-511). Springer, Berlin, Heidelberg.

[34] Chandu, R., Khyathi, Chinnakotla, M., Black, A. W., & Shrivastava, M. (2017). WebShodh: A Code Mixed Factoid Question Answering System for Web. In *International Conference of the Cross-Language Evaluation Forum for European Languages* (pp. 104-111). Springer, Cham.

[35] Choudhary, B., & Bhattacharyya, P. (2002). Text clustering using universal networking language representation. In *Proceedings of Eleventh International World Wide Web Conference*.

[36] Choudhury, M., Ershadul, H., Nawab, Y.A., Mohammad, Z.H.S., & Ahsan, R.M. (2005). Bridging Bangla to Universal Networking Language-a human language neutral meta-language. In *International Conference on Computer and Information Technology (ICCIT), Dhaka* (pp. 104-109).

[37] Concepts of UW [Online]. Available: http://www.undl.org/unlsys/unl/unl2005/uw.htm

[38] Cooper, R. J., & Ruger, S. M. (2000). A simple question answering system. In *TREC- 9*.

[39] Dave, S., & Bhattacharyya, P. (2001). Knowledge extraction from Hindi text. *IETE Technical Review*, *18*(4), 323-331.

[40] Dey, K., & Bhattacharyya, P. (2005). Universal Networking Language based analysis and generation of Bengali case structure constructs. *Universal Network Language: Advances in Theory and Applications*, *12*, 215-229.

[41] Dhanabalan, T., & Geetha, T. V. (2003). UNL deconverter for Tamil. In *International Conference on the Convergences of Knowledge, Culture, Language and Information Technologies*.

[42] Dhanabalan, T., Saravanan, K., & Geetha, T. V. (2002). Tamil to UNL EnConverter. In *Proc. Int. Conf. on Universal Knowledge and Language* (pp. 1-16).

[43] Dictionary Specifications, [Online].
Available: http://www.unlweb.net/wiki/Dictionary_Specs

[44] Di. W., Boytsov, L., Araki, J., Patel, A., Gee, J., Liu, Z., Nyberg, E., & Mitamura, T. (2014). CMU Multiple-choice Question Answering System at NTCIR-11 QA-Lab. In *NTCIR*.

[45] Document Types [Online]. Available:
http://www.facweb.iitkgp.ernet.in/~sudeshna/courses/nlp07/lec24-QA.pdf

[46] Mohnish, D., Banerjee, D., Chaudhuri, D., & Lehmann, J. (2018). EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs. *arXiv preprint arXiv:1801.03825*.

[47] EUGENE [Online].
Available: http://dev.undlfoundation.org/generation/index.jsp

[48] Feng, G., Du, Z. & Wu, X. (2018). A Chinese Question Answering System in Medical Domain. *Journal of Shanghai Jiaotong University (Science)*, Vol 23, 678-683.

[49] Fourth Olympiad [Online].
Available: http://www.unlweb.net/wiki/IV_UNL_Olympiad

[50] Frederik, S., Schüler, R., Draeger, T., Dummer, D., Ernst, A., Flemming, P., & Neves, M. (2016). Hpi question answering system in bioasq 2016. In *Proceedings of the Fourth BioASQ workshop* (pp. 38-44).

[51] Gaizauskas, R., & Humphreys, K. (2000). A combined IR/NLP approach to question answering against large text collections. In *Content-Based Multimedia Information Access-Volume 2* (pp. 1288-1304). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[52] Gillard, L., Bellot, P., & El-Beze, M. (2006). Question answering evaluation survey. In *Language Resources and Evaluation Conference*.

[53] Gill, M.S. (2008). D*evelopment of a Punjabi grammar checker* (Doctoral dissertation), Punjabi University, Patiala, India.

[54] Goel, K. (2016). *Information Retrieval System using UNL for Multilingual Question Answering* (M.E dissertation), Thapar University, Patiala.

[55] Green, F., Bert, Alice, K., Wolf, Chomsky, C., & Laughery, K. (1963). Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference* (pp. 219-224). ACM.

[56] Gupta, D., Kumari, S., Ekbal, A., & Bhattacharyya, P. (2018). MMQA: A Multi-domain Multi-lingual Question-Answering Framework for English and Hindi. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*.

[57] Gupta, D., Lenka, P., Ekbal, A., & Bhattacharyya, P. (2018). Uncovering Code-Mixed Challenges: A Framework for Linguistically Driven Question Generation and Neural based Question Answering. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*. 119-130.

[58] Hajlaoui, N., & Boitet, C. (2005). A" pivot" XML-based architecture for multilingual, multiversion documents: parallel monolingual documents aligned through a central correspondence descriptor and possible use of UNL. *Research on Computing Science*, *12*, 309-326.

[59] Harabagiu, S., & Moldovan, D. (2003). Question answering. In Mitkov, R. (Ed.), *The Oxford Handbook of Comp. Linguistics*, chapter 31, pp. 560–582. Oxford University Press.

[60] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Girju, R., Rus, V., & Morarescu, P. (2000). FALCON: Boosting Knowledge for Answer Engines. In *TREC* (Vol. 9, pp. 479-488).

[61] Hartawan, A., & Suhartono, D. (2015). Using vector space model in question answering system. *Procedia Computer Science*, *59*, 305-311.

[62] Heba, A., Maha, R., Bassant, F., Reham, M., Alaa, M., Nagwa, E.M., & Marwan, R. (2014). Al-Bayan: an Arabic question answering system for the Holy Quran. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)* (pp. 57-64).

[63] Helena, G.A., Grigori, S., Darnes, V., & David, P. (2015). CICBUAPnlp: Graph-Based Approach for Answer Selection in Community Question Answering Task. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 18-22).

[64] Hermjakob, U. (2001). Parsing and question classification for question answering. In *Proceedings of the workshop on Open-domain question answering-Volume 12* (pp. 1-6). Association for Computational Linguistics.

[65] Hirschman, L., & Gaizauskas, R. (2001). Natural language question answering: the view from here. *natural language engineering*, *7*(4), 275-300.

[66] Hua, H.U., & Shi-zheng, Z. (2017). Design of Intelligent Tourism Question Answering System Based on Semantic Web. *DEStech Transactions on Economics, Business and Management*, (eced).

[67] Hyeon-gu, L., Kim, M., Kim, H., Kim, J., Kwon, S., Seo, J., & Choi, J. K. (2016). KSAnswer: question-answering system of Kangwon national university and Sogang university in the 2016 BioASQ challenge. In *Proceedings of the Fourth BioASQ workshop* (pp. 45-49).

[68] IAN [Online]. Available: http://dev.undlfoundation.org/analysis/index.jsp

[69] Interactive Analyzer [Online]. Available: http://www.unlweb.net/wiki/IAN

[70] Iraola, L. (2005). Using WordNet for linking UWs to the UNL UW System. *Universal Networking Language: Advances in Theory and Applications*, 370.

[71] Iyer, J. A., & Bhattacharyya, P. (2005). Using semantic information to improve case retrieval in case-based reasoning systems. *Universal Networking Language: Advances in Theory and Applications*, 347.

[72] Jadhav, N., & Bhattacharyya, P. (2014). Dive deeper: deep semantics for sentiment analysis. In *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis* (pp. 113-118).

[73] Jain, M., & Damani, O. P. (2009). English to UNL (Interlingua) enconversion. In *Proc. Second Conference on Language and Technology,(CLT)*.

[74] Jiang, C., Tissiani, G., Falquet, G., & Rodolfo, P. (2005). Facilitating communication between languages and cultures: a computerized interface and knowledge base. *Universal Network Language: Advances in Theory and Applications*, *12*, 358-368.

[75] Jun, Y., Xin, J., Zhengdong, L., Lifeng, S., Hang, L., & Xiaoming, L. (2016). Neural generative question answering. *arXiv preprint arXiv:1512.01337*.

[76] Junwei, B., Duan, N., Zhou, M., & Zhao, T. (2014). Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Vol. 1, pp. 967-976).

[77] Kangavari, M. R., Ghandchi, S., & Golpour, M. (2008). Information retrieval: Improving question answering systems by query reformulation and answer validation. *World Academy of Science, Engineering and Technology*, *48*, 303-310.

[78] Karande, J. B. (2007). Multilingual search engine: implementation using UNL. In *Proceedings of International Conference on Semantic Web and Digital Libraries, Bangalore, India* (pp. 1-7).

[79] Kotaro, S., Hyogo, M., Eisuke, M., Takahisa, J., Hideyuki, S., Tatsunori, M., Madoka, I., & Noriko, K. (2014). Forst: Question Answering System Using Basic Element at NTCIR-11 QA-Lab Task. In *NTCIR*.

[80] Kotaro, S., Takaaki, M., Madoka, I., Hideyuki, S., Tatsunori, M., Noriko, K., & Teruko, M. (2017). FelisCatusZero: A world history essay question answering system for the University of Tokyo's entrance exam. In *Proceedings of Open Knowledge Base and Question Answering Workshop at SIGIR, Tokyo, Japan.*

[81] Kumar, P. (2012). *UNL Based Machine Translation System for Punjabi Language* (Doctoral dissertation), Thapar University, Patiala.

[82] Kumar, P., & Sharma, R. K. (2013). Punjabi DeConverter for generating Punjabi from Universal Networking Language. *Journal of Zhejiang University Science C*, *14*(3), 179-196.

[83] Kumar, P., & Sharma, R. K. (2012). Punjabi to UNL enconversion system. *Sadhana*, *37*(2), 299-318.

[84] Kwong, S.F., Chih, H.B. (2017). A Hybrid Question Answering System based on Ontology and Topic Modeling. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9*(2-10), 151-158.

[85] Lafourcade, M. (2005). Semantic analysis through ant algorithms, conceptual vectors and fuzzy UNL graphs. *Universal Network Language: Advances in Theory and Applications*, *12*, 125-137.

[86] Lafourcade, M., & Boitet, C. (2002). UNL Lexical Selection with Conceptual Vectors. In *LREC*.

[87] Lekshmi, R.P., Veena, G., & Deepa, G. (2018). A Combined Approach Using Semantic Role Labelling and Word Sense Disambiguation for Question Generation and Answer Extraction. In *2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC)* (pp. 1-6). IEEE.

[88] Lehnert, W. G. (1978). *The process of question answering: A computer simulation of cognition* (Vol. 978). Hillsdale, NJ: Lawrence Erlbaum.

[89] Lin, Q., Zhou, H., Qu, Y., Zhang, W., Li, S., Rong, S., & Yu, Y. (2018). QA4IE: A Question Answering based Framework for Information Extraction. *arXiv preprint arXiv:1804.03396*.

[90] Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1-7). Association for Computational Linguistics.

[91] Malinowski, M., & Fritz, M. (2014). A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in neural information processing systems* (pp. 1682-1690).

[92] Manuel, E.S., and Riofrio, G.E. (2010). Architecture of a Question-Answering System for a specific repository of documents. In *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on* (Vol. 2, pp. V2-12). IEEE.

[93] Martins, R. (2003). HOW many colors should be in the rainbow? Representing color names in knowledge bases. In *International Conference on the Convergence of Knowledge, Culture, Language and Information Technologies, Alexandria, Egypt*.

[94] Martins, R., Hasegawa, R., Graças, M., & Nunes, V. (2005). Hermeto: a NL–UNL enconverting environment. *Universal Network Language: Advances in Theory and Applications*, *12*, 254-260.

[95] Martins, RT. (2002). The UNL distinctive features: evidences through a NL-UNL encoding task. In *The First International Workshop on UNL, other Interlinguas and their Applications. LREC 2002, May*.

[96] Michele, B., Eric. B., Susan, D., & Jimmy, L. (2002). Askmsr: Question answering using the worldwide web. In *Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases* (pp. 7-9).

[97] Miguel, R., Falé, M., & Couto, F. (2016). WS4A: a biomedical question and answering system based on public web services and ontologies. *arXiv preprint arXiv:1609.08492*.

[98] Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39-41.

[99] Mridha, M. F., Hossain, M. Z., & Noor, S. A. (2010). Development of morphological rules for bangla words for universal networking language. *Int. J. of Computer Science and Network Security*, *10*(10), 231-237.

[100] Mridha, M. F., Saha, A. K., & Das, J. K. (2014). New approach of solving semantic ambigssuity problem of Bangla Root words using universal networking language (UNL). In *Informatics, Electronics & Vision (ICIEV), 2014 International Conference on* (pp. 1-6). IEEE.

[101] Mohamed, A.H.A. (2017). *An automatic question answering system for the Arabic Quran* (Doctoral dissertation, Sudan University of Science and Technology).

[102] Mohanty, R., Dutta, A., & Bhattacharyya, P. (2005). Semantically relatable sets: building blocks for representing semantics. In *MT Summit* (Vol. 5).

[103] Mohanty, R. K., Almeida, A. F., & Bhattacharyya, P. (2005). Prepositional phrase attachment and interlingua. *Research on Computing Science Volume 12*, 241-253.

[104] Moldovan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Goodrum, R., Girju, R., & Rus, V. (1999). Lasso: A tool for surfing the answer net. In *TREC* (Vol. 8, pp. 65-73).

[105] Montesco, C.E., Moreira, D.A. (2005).UCL—Universal Communication Language. *Universal Networking Language: Advances in Theory and Applications*, 326.

[106] Muhammad, Z. A., & Nadeem, N. (2018). Convolutional Neural Network: Text Classification Model for Open Domain Question Answering System. *arXiv preprint arXiv:1809.02479*.

[107] Mukerjee, A., Raina, A. M., Kapil, K., Goyal, P., & Shukla, P. (2003). Universal Networking Language: a tool for language independent semantics?. *Universal Networking Language: Advances in Theory and Applications*, 145.

[108] NABEEL, N., & SAIDAH, S. (2017). QUESTION ANSWERING SYSTEM SUPPORTING VECTOR MACHINE METHOD FOR HADITH DOMAIN. *Journal of Theoretical & Applied Information Technology*, *95*(7).

[109] Navaneethakrishnan, S. C., & Parthasarathi, R. (2015). Building a Language-Independent Discourse Parser using Universal Networking Language. *Computational Intelligence*, *31*(4), 593-618.

[110] Nguyen, D. P., & Ishizuka, M. (2006). A statistical approach for universal networking language-based relation extraction. In *RIVF* (pp. 153-160).

[111] Nicosia, M., Filice, S., Barrón-Cedeno, A., Saleh, I., Mubarak, H., Gao, W., & Màrquez, L. (2015). QCRI: Answer selection for community question answering-experiments for Arabic and English. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 203-209).

[112] Oren, E., Fader, A., Christensen, J., Soderland, S., & Mausam, M. (2011, July). Open information extraction: The second generation. In *IJCAI* (Vol. 11, pp. 3-10).

[113] Park, S., Kwon, S., Kim, B., & Lee, G. G. (2015). ISOFT at QALD-5: Hybrid Question Answering System over Linked Data and Text Data. In *CLEF (Working Notes)*.

[114] Pelizzoni, J. M., & Nunes, M. D. G. V. (2005). Flexibility, configurability and optimality in UNL deconversion via multiparadigm programming. *Universal Networking Language: Advances in Theory and Applications*, *12*, 175-194.

[115] Prager, J., Radev, D., Brown, E., & Coden, A. (1999). The use of predictive annotation for question answering in TREC8. In *NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC 8)*.

[116] Pragisha, K., & Reghuraj, D. P. (2014). A Natural Language Question Answering System in Malayalam Using Domain Dependent Document Collection as Repository. *International Journal of Computational Linguistics and Natural Language Processing*, *3*(3), 2279-0756.

[117] Pratap, U., & Srivastava, R. K. (2013). Transforming business with mobile enterprise apps. *International Journal of Engineering and Computer Science*, *2*(06).

[118] Punjabi Language [Online]. Available: http://en.wikipedia.org/wiki/Punjabi_language

[119] Quan, H. T., Tran, V., Vu, T., Nguyen, M., & Pham, S. B. (2015). JAIST: Combining multiple features for answer selection in community question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 215-219).

[120] Radev, D., Fan, W., Qi, H., Wu, H., & Grewal, A. (2002). Probabilistic question answering on the web. In *Proceedings of the 11th international conference on World Wide Web* (pp. 408-419). ACM.

[121] Ramamritham, K., Bahuman, A., & Duttagupta, S. (2006). aAqua: a database-backended multilingual, multimedia community forum. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* (pp. 784-786). ACM.

[122] Ravichandran, D., & Hovy, E. (2002). Learning surface text patterns for a question answering system. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 41-47). Association for Computational Linguistics.

[123] Ribeiro, C., Santos, R., Chaves, R. P., & Marrafa, P. (2004). Semi-Automatic UNL Dictionary Generation Using WordNet. PT. In *Proceedings of 4th International Conference on Language Resources and Evaluation,* Lisbon, pp. 279-282.

[124] Rouquet, D., & Nguyen, H. (2009). Interlingual annotation of texts in the OMNIA project. In *Proceedings of 4th International Conference on Language and Technology*, Poland, pp. 1-5.

[125] Roy, A. T., Islam, M. A., & Ali, M. N. Y. (2016). Templates of Dictionary Entries of Bangla Repetition Words for Universal Networking Language. *International Journal of Reliability, Quality and Safety Engineering*, *23*(03), 1640003.

[126] Sathiyamurthy, K., Panimalar, D., & Pandian, S. L. (2014). Multilingual acquiring of e-content definition based on universal networking language. In *MOOC, Innovation and Technology in Education (MITE), 2014 IEEE International Conference on* (pp. 240-244). IEEE.

[127] Scope and goals of UNL [Online]. Available: http://www.unlweb.net/unlweb/index.php?option=com_content&view=article&id=58:unl&catid=54:unlweb

[128] Sean, G., Zadrozny, W., & Avadhani, A. (2014). Watsonsim: Overview of a question answering engine. *arXiv preprint arXiv:1412.0879*.

[129] Second Olympiad. 2013. [Online]. Available: http://www.unlweb.net/wiki/II_UNL_Olympiad

[130] Sérasset, G., & Boitet, C. (2000). On UNL as the future html of the linguistic content & the reuse of existing NLP components in UNL-related applications with the example of a UNL-French deconverter. In *Proceedings of the 18th conference on Computational linguistics-Volume 2* (pp. 768-774). Association for Computational Linguistics.

[131] Shih, C. W., Day, M. Y., Tsai, T. H., Jiang, T. J., Wu, C. W., Sung, C. L., & Hsu, W. L. (2005). ASQA: Academia sinica question answering system for NTCIR-5 CLQA. In *NTCIR-5 Workshop, Tokyo, Japan* (pp. 202-208).

[132] Shizhu, H., Zhang, Y., Liu, K., & Zhao, J. (2014, September). CASIA@ V2: A MLN-based Question Answering System over Linked Data. In *CLEF (Working Notes)* (pp. 1249-1259).

[133] Shi, X., Chen, Y., Cardenas, J., Gelbukh, A., & Tovar, E. M. (2005). A UNL DeConverter for Chinese Universal Network Language. *Universal Network Language: Advances in Theory and Applications*, 167-174.

[134] Shukla, P., Mukherjee, A., & Raina, A. (2004). TOWARDS A LANGUAGE INDEPENDENT ENCODING OF DOCUMENTS. *NLUCS 2004*, 116.

[135] Singh, H., & Bhatia, P. (2013). *NLization of adjectives conjunctions determiners and verbs with EUGENE* (M.E dissertation), Thapar University, Patiala.

[136] Singh, H., & Kumar, P. (2013). Implementation of NLization framework for verbs, pronouns and determiners with Eugene. *International Journal on Natural Language Computing (IJNLC)*, 2(3), 73-85.

[137] Singh, H., & Kumar, P. (2013). Analysis of Noun, Pronoun, and Adjective Morphology for NLIzation of Punjabi with EUGENE. *International Journal of Computational Linguistics and Natural Language Computing (IJCLNLC)*, 2(7), 436-442.

[138] Sleator, D., & Temperley, D. (1991). Parsing English with a link grammar. *arXiv preprint cmp-lg/9508004*.

[139] Soderland, S., Roof, B., Qin, B., Xu, S., Mausam, & Etzioni, O. (2010). Adapting open information extraction to domain-specific relations. *AI magazine*, *31*(3), 93-102.

[140] Sommer, R., & Paxson, V. (2010). Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *IEEE Symposium on Security and Privacy*, Berkeley/Oakland, CA, 305-316.

[141] Sornlertlamvanich, V., Potipiti, T., & Charoenporn, T. (2001). UNL document summarization. In *Proceedings of the First International Workshop on Multimedia Annotation*.

[142] Sridhar, R., Sethuraman, P., & Krishnakumar, K. (2016). English to Tamil machine translation system using universal networking language. *Sādhanā, 41*(6), 607-620.

[143] Stoyanchev, S., Song, YC., and Lahti, W. 2008. Exact phrases in information retrieval for question answering. *In Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pp. 9-16.

[144] Stupina, A., Zhukov, E.A., Ezhemanskaya, S., Karaseva, M. V., & Korpacheva, L. N. (2016, November). Question-answering system. In *IOP Conference Series: Materials Science and Engineering* (Vol. 155, No. 1, p. 012024). IOP Publishing.

[145] Subramanian, R., & Narayanan, A. (1990). An AI-based approach to machine translation in Indian languages. *Communications of the ACM*, *33*(5), 521-527.

[146] Sujata, R., & Parteek, K. (2017). A sentiment analysis system to improve teaching and learning. *IEEE Comput*, 36-43.

[147] Surve, M., Singh, S., Kagathara, S., Venkatasivaramasastry, K., Dubey, S., Rane, G., Saraswati, J., Badodekar, S., Iyer, A., Almeida, A., Nikam, R., Perez, CG., and Bhattacharyya, P. (2004). Agro-explorer: A meaning based multilingual search engine. In *International Conference on Digital Libraries (ICDL)*.

[148] Takaaki, M., Francesco, C., Fadi, B., Evan, C., Cheng-Ta, C., Keyang, X, and Teruko, M. (2017). Wikipedia Based Essay Question Answering System for University Entrance Examination. In *Proceedings of Open Knowledge Base and Question Answering Workshop at SIGIR*.

[149] Tarcisio, D. S., & Nagi, M. 2008. Sharing knowledge across language barriers: a universal approach for online books. In *Proceedings of the 2008 ACM workshop on Research advances in large digital book repositories* (pp. 49-52).

[150] Third Olympiad. 2014. [Online]. Available: http://www.unlweb.net/wiki/III_UNL_Olympiad

[151] Tomokiyo, M., & Chollet, G. (2003). VoiceUNL: a proposal to represent speech control mechanisms within the Universal Networking Digital Language. In *International Conference on the Convergence of Knowledge, Culture, Language and Information Technologies, Egypt*.

[152] Transformation Rules, [Online]. Available: http://www.unlweb.net/wiki/T-rule

[153] Uchida, H. (2005). UNL: Universal Networking Language. *http://www. undl. org/*.

[154] Uchida, H. (1987). ATLAS: Fujitsu machine translation system. *Proc. Machine Translation Summit, Japan*, 17-19.

[155] Uchida, H., Zhu, M. (2005). UNL for providing knowledge infrastructure. *Proceedings of Semantic Computing Workshop*, Chiba, Japan.

[156] Uchida, H., & Zhu, M. (2001). The universal networking language beyond machine translation. In *International Symposium on Language in Cyberspace, Seoul* (pp. 26-27).

[157] Uchida, H., & Zhu, M. (1993). Interlingua for multilingual machine translation. *Proceedings of MT Summit IV, Kobe, Japan*, 157-169.

[158] Uchida, H., Zhu, M., & Della Senta, T. (1999). The UNL, A gift for a millennium. *IAS/UNU, Tokyo*.

[159] UNDL foundation, (2013) [Online].
Available: http://www.undlfoundation.org

[160] UNL Web. (2013).[Online].
Available: http://www.unlweb.net/user/index.php?page-login

[161] UNL wiki. (2013) [Online].
Available: http://www.unlweb.net/wiki

[162] Vasileios. L., Passalis, N., and Tefas. A. (2018). Visual Question Answering using Explicit Visual Attention. In *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on* (pp. 1-5). IEEE.

[163] Verma, A. & Bhatia, P. (2013). NLization Framework with EUGENE. *International Journal on Natural Language Computing (IJNLC)*, *2*(2).

[164] Verma, A., & Bhatia, P. (2013). *NLization of Nouns, Pronouns, Prepositions and Sentence Structures with EUGENE* (M.E dissertation), Thapar University, Patiala.

[165] Verma, A., & Kumar, P. (2013). NLization of nouns, pronouns and prepositions in Punjabi With EUGENE. *International Journal on Natural Language Computing (IJNLC)*, *2*(2).

[166] Verma, N., & Bhattacharyya, P. (2005). Automatic generation of multilingual lexicon by using WordNet. In *International Conference on Convergence of Knowledge, Culture, Language and Information Technology. Library of Alexandria, Egypt*.

[167] Waheeb, A., & Babu, A. (2017). An automatic web-based question answering system for e-learning. *Information Technology and Learning Tools*, (58, Issue. 2), 1-10.

[168] Wang, D., & Nyberg, E. (2015). A long short-term memory model for

answer sentence selection in question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (Vol. 2, pp. 707-712).

[169] Wanyun, C., Yanghua, X., Haxiun, W., Yangqiu, S., Seung-won, H., & Wei, W. (2017). KBQA: learning question answering over QA corpora and knowledge bases. *Proceedings of the VLDB Endowment*, *10*(5), 565-576.

[170] Witten, I. H., Moffat, A., & Bell, T. C. (1999). *Managing gigabytes: compressing and indexing documents and images*. Morgan Kaufmann.

[171] Woods, W. A. (1973). Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition* (pp. 441-450). ACM.

[172] X-Bar theory. [Online].
Available: http://www.unlweb.net/wiki/X-bar_theory

[173] Xiaoyi, Y., Xie, J., & Liu, Y. (2017). Implementation of the Intelligent Question Answering System Based on Remote Service Framework. *DEStech Transactions on Social Science, Education and Human Science*, (icsste).

[174] Xiong, C., Merity, S., & Socher, R. (2016, June). Dynamic memory networks for visual and textual question answering. In *International conference on machine learning* (pp. 2397-2406).

[175] Xu, J., Licuanan, A., & Weischedel, R. M. (2003). TREC 2003 QA at BBN: Answering Definitional Questions. In *TREC-12* (pp. 98-106).