

WORD PHRASE ALIGNMENT

*in partial fulfillment of the requirements for the award of the degree
of Bachelor of Engineering in*

INFORMATION TECHNOLOGY

Submitted by

**GAURAV BHISIKAR
ADITYA DALE
SAGAR MALIK**

Under the guidance of

PROF. POOJA WALKE

Assistant Professor

Academic Year 2020-21

Department of Information Engineering



**ST. VINCENT PALLOTTI COLLEGE OF
ENGINEERING AND TECHNOLOGY**

Wardha Road, Gavsi Manapur, Nagpur

**ST. VINCENT PALLOTTI COLLEGE OF
ENGINEERING AND TECHNOLOGY -NAGPUR**

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

Certified that this project report “**WORD PHRASE ALIGNMENT**” is the bonafide work of “**GAURAV BHISIKAR, SAGAR MALIK, ADITYA DALE**” who carried out the project work under my supervision in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in **INFORMATION TECHNOLOGY** of RASHTRASANT TUKADOJI MAHARAJ NAGPUR UNIVERSITY, NAGPUR

DR. MANOJ V. BRAMHE

Associate Professor

HEAD OF THE DEPARTMENT

PROF. POOJA WALKE

Assistant Professor

GUIDE

PRINCIPAL

**ST. VINCENT PALLOTTI COLLEGE OF ENGINEERING AND
TECHNOLOGY**



International Institute of Information Technology
A Research University

Date –15/06/2021

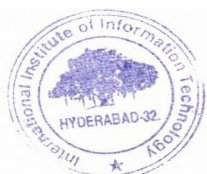
To Whom It May Concern

This is to certify that **Mr. Gaurav Bhisikar** of the Information Technology Department from St Vincent Pallotti College of Engineering and Technology, Nagpur did his Final year Major project of the academic year 2020-21. He successfully assisted on Research (duration:- 20th July 2020 – 10th June 2021) Work at Anusaaraka Lab, in LTRC on the topic “**Word Phrase Alignment**”.

His contribution has been satisfactory over the said duration.

Vineet Chaitanya

(Vineet Chaitanya)
Distinguished Professor





International Institute of Information Technology
A Research University

Date –15/06/2021

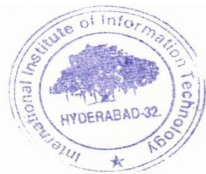
To Whom It May Concern

This is to certify that **Mr. Aditya Dale** of the Information Technology Department from St Vincent Pallotti College of Engineering and Technology, Nagpur did his Final year Major project of the academic year 2020-21. He successfully assisted on Research (duration:- 20th July 2020 – 10th June 2021) Work at Anusaaraka Lab, in LTRC on the topic “**Word Phrase Alignment**”.

His contribution has been satisfactory over the said duration.

Vineet Chaitanya

(Vineet Chaitanya)
Distinguished Professor





International Institute of Information Technology
A Research University

Date –15/06/2021

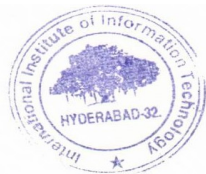
To Whom It May Concern

This is to certify that **Mr. Sagar Malik** of the Information Technology Department from St Vincent Pallotti College of Engineering and Technology, Nagpur did his Final year Major project of the academic year 2020-21. He successfully assisted on Research (duration: - 20th July 2020 – 10th June 2021) Work at Anusaaraka Lab, in LTRC on the topic “**Word Phrase Alignment**”.

His contribution has been satisfactory over the said duration.

Vineet Chaitanya

(Vineet Chaitanya)
Distinguished Professor



ACKNOWLEDGEMENT

Our major project is titled, “**WORD PHRASE ALIGNMENT**”. Any project requires a lot of hard work, sincerity, and systematic work methodologies. We express our deepest gratitude to our Project Guide, **Prof. Pooja Walke**, for giving us an opportunity to be a part of this project and for the guidance, provided throughout the span of our project.

We owe our gratitude to the industry mentor **Dr. Soma Paul**, research Assistant professor, **IIT Hyderabad**, and **Dr. Vineet Chaitanya at Language Translation Research Center (LTRC), IIT Hyderabad** for their immense support and guidance throughout the project. Our special thanks to **Ms. CBS Manaswini**, an alumni mentor, for her proactive guidance on the student projects.

We would also like to thank **Dr. Manoj V. Bramhe, Head of the Department of Information Technology**, and all our faculty members who regularly evaluated our major project and pointed out the shortcomings in the project. They also gave us important feedback for the further improvement of our project. We are highly indebted to them.

We are also grateful to the **Management of the College, Dr. Surendra V. Gole, Principal & Prof. R. B. Gowardhan, Vice-Principal** for the overwhelming support in providing us the online facilities and other required resources. We would like to thank our Library Department for providing us useful books and research articles related to our project in online mode.

Project Group Members

GAURAV BHISIKAR

SAGAR MALK

ADITYA DALE

ABSTRACT

The topic for our major project is “Word-Phrase Alignment”. The main objective of this project is to develop software tools and a scheme for building a word or phrase-aligned corpus of English and Hindi optimally using existing open resources. NMT models that are used to translate the English language to Hindi language train on a corpus which is on sentence-level so during training they learn to translate on a sentence level, as a result, the current NMT models are not able to generate correct alignment of translated Hindi sentence. So this project focuses on identifying the relationship between words of both English and Hindi sentences and to develop an aligned parallel corpus on word/phrase level.

Word alignment is the natural language processing task of identifying translation relationships among the words (or more rarely multiword units) in a bitext, resulting in a bipartite graph between the two sides of the bitext, with an arc between two words if and only if they are translations of one another. Word alignment is typically done after sentence alignment has already identified pairs of sentences that are translations of one another.

Bitext word alignment is an important supporting task for most of the methods of statistical machine translation. The parameters of statistical machine translation models are typically estimated by observing word-aligned bitexts, and conversely, automatic word alignment is typically done by choosing the alignment that best fits a statistical machine translation model. The circular application of these two ideas results in an instance of the expectation-maximization algorithm.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	i
	TABLE OF CONTENT	ii
	ABSTRACT	iii
	LIST OF FIGURES	iv
1.	INTRODUCTION	1-2
	1.1 OVERVIEW	1
	1.2 PROBLEM STATEMENT	1
	1.3 OBJECTIVE	2
	1.4 ORGANIZATION OF REPORT	2
2.	REVIEW OF LITERATURE	3-6
	2.1 INTRODUCTION	3
	2.2 LITERATURE SURVEY	3
	2.3 FEASIBILITY STUDY	5
	2.3.1 TECHNICAL FEASIBILITY 2.3.2 ECONOMICAL FEASIBILITY 2.3.3 OPERATIONAL FEASIBILITY 2.3.4 SCHEDULE FEASIBILITY	
3.	PROPOSED SYSTEM	7-21
	3.1 DRAWBACK OF CURRENT SYSTEM AND NEED OF PROPOSED SYSTEM	7
	3.2 SYSTEM DESCRIPTION AND SRS	7
	3.2.1 PURPOSE 3.2.2 SCOPE OF PROJECT 3.2.3 PROJECT PLANNING	

	3.2.4 WORK BREAK DOWN STRUCTURE	
	3.3 SYSTEM ANALYSIS 3.3.1 SYSTEM ARCHITECTURE 3.3.2 FLOWCHART OF SYSTEM 3.3.3 DATA FLOW DIAGRAM 3.3.4 USE CASE DIAGRAM 3.3.5 HARDWARE REQUIREMENTS 3.3.6 SOFTWARE REQUIREMENTS	11
	3.4 SYSTEM DESIGN 3.4.1 CLASS DIAGRAM 3.4.2 ACTIVITY DIAGRAM 3.4.3 COMPONENT DIAGRAM 3.4.4 DEPLOYMENT DIAGRAM	17
4	IMPLEMENTATION AND CODING	22-33
	4.1 SYSTEM RELEASE PLAN	22
	4.2 IMPLEMENTATION METHODOLOGY	24
	4.3 CODE AND SCREENSHOTS	30
5	TESTING	34-37
	5.1 INTRODUCTION	34
	5.2 FUNCTIONALITY TESTING	34
	5.3 UNIT TESTING	36
	5.4 INTEGRATION TESTING	37
6	CONCLUSION AND FUTURE SCOPE	38
	6.1 CONCLUSION	38
	6.2 FUTURE SCOPE	38
7	REFERENCES	39-40
8	ANNEXURE I: Stockholder Detail	41-42
9	PROJECT SPECIFIC PO MAPPING	43-44

LIST OF FIGURES

Sr. No.	FIG. NO.	TITLE	PAGE NO.
1	3.2.3	GANTT CHART	9
2	3.3.1	SYSTEM ARCHITECTURE	12
3	3.3.2	FLOWCHART	13
4	3.3.3.1	DFD LEVEL 0	14
5	3.3.3.2	DFD LEVEL 1	15
6	3.3.4	USECASE DIAGRAM	16
7	3.4.1	CLASS DIAGRAM	18
8	3.4.2	ACTIVITY DIAGRAM	19
9	3.4.3	COMPONENT DIAGRAM	20
10	3.4.4	DEPLOYMENT DIAGRAM	21
11	4.2.1.1	INPUT OF REMOVE NOISE	24
12	4.2.1.3	OUTPUT OF REMOVE NOISE	25
13	4.2.2.1	INPUT OF SENT2TABLE	26
14	4.2.2.2	OUTPUT OF SENT2TABLE	27
15	4.2.3.1	INPUT OF PARSER-MODIFICATION	28
16	4.2.3.2	OUTPUT OF PARSER-MODIFICATION	29
17	4.3.1	CODE SNAPSHOT (Sent2table.py)	30
18	4.3.2	CODE SNAPSHOT (removeNoise.py)	32
19	5.2	FUNCTIONALITY TESTING	35
20	5.3	UNIT TESTING	36
21	5.4	INTEGRATION TESTING	37

1. INTRODUCTION

1.1 OVERVIEW

This is a research project in the domain of NLP, at Anusaarka Lab under the LTRC center at IIIT Hyderabad. This project aims to modify the current method of training deep learning models from sentence level to word/phrase level. The Current method of training deep learning model for translation uses parallel corpus (i.e. a parallel corpus is a set formed by a text and its translation) which is on a sentence level. But this method requires a very large corpus to train the model which makes it difficult to get expected results from a smaller set of data. So if we can convert the parallel corpus to word/phrase structure level it will be easier to train and get a better result from a deep learning model.

So comparatively model will require a small-sized corpus for training, eventually, it will reduce the time to train a model, as the size of data is reduced. It will also give more human control over the traditional model.

Above mentioned is the ultimate goal of the research work. But on this project, we are working on the possibility of automating the task of alignment of words/phrases of the translated sentence in order to map these words/phrases with the words/phrases with the same meaning in the source sentence i.e. to create a parallel corpus with word/phrase level structure.

1.2 PROBLEM STATEMENT

To develop a software tool and a scheme for building a word/phrase aligned corpus of English and Hindi using existing open resources in an optimal manner.

1.3 OBJECTIVE

The goal of this project is to develop software tools for Word Phrase Alignment. Our Project focuses on extracting translation information from a parallel text and determining which words in the source text correspond to which words in the target text.

1.4 ORGANIZATION OF REPORT

Our report is divided into seven chapters which include an introduction, review of literature, proposed system, implementation, testing, conclusion, and reference. Chapter 1 includes the complete introduction, chapter 2 consists of the review of the literature that discusses what has been derived from papers, online forums and what previous work is done, and the need of the proposed system. We have analyzed the various terms related to Alignment of phrases such as parallel text, Parallel Text Alignment, Alignment Details, etc. Chapter 3 consists of the proposed system that describes the desired features and functions of the system. System analysis and design with the help of class diagrams and component diagrams have been explained in detail. Hardware and software requirements are also mentioned in this chapter. Chapter 4 consists of our implementation methodologies, system release plan, and codes and its snapshots. Chapter 5 contains our test cases, testing of various tools, testing, and analysis of outputs of tools and scripts we wrote. In chapter 6 conclusion and future scope is given. Chapter 7 consists of all the references we used during our project.

2. REVIEW OF LITERATURE

2.1 INTRODUCTION

A literature review is an evaluative report of information found in the literature related to your selected area of study. The review should describe, summarize, evaluate and clarify this literature. It should give a theoretical base for the research and help you to determine the nature of your research.

When conducting research, a literature review is an essential part of the project because it covers all previous research done on the topic and sets the platform on which the current research is based. No new research can be taken seriously without first reviewing the previous research done on the topic.

2.2 LITERATURE SURVEY

In paper [1] Srivastava, Jyoti, and Sudip Sanyal describe the use of POS tag for enhance the performance of statistical word alignment. The approach shown in this paper works with a small size of the corpus. The authors conducted Experiments on a corpus of 1000 sentences taken from TDIL sample tourism, for the English-Hindi language pair. 950 sentences Out of 1000 sentences were used for training and the remaining 50 sentences were used for testing.

In paper [2] the authors conducted a survey on Performance Improvement of Word Phrase Alignment in English to Hindi Translation System. In this paper, the authors reviewed and discussed different types of techniques of word alignment for the English to Hindi translation system. This paper uses a combined technique to progress the implementation of word alignment for the English-Hindi language pair with limited resources.

In paper [3] Santanu Pal, Aniruddha Gosh, and Sivaji Bandyopadhyay, discuss the Word Alignment between English and Bengali language, for this, the author of [3] used the approach of bootstrapping method in Phrase-Based Machine Translation. The objective of this paper is to analyze the effects of word alignment in parallel English – Bengali corpus in a Phrase-based Statistical Machine Translation system. The rule-based word mapping system uses the English WordNet1 to analyze the morphological information of English words and a morphological analyzer to analyze the morphology information of Bengali words. In this paper, a rule-based and a statistical model, have been implemented for the word alignment system. Data-preprocessing plays a crucial role in Phrase-based Statistical Machine Translation (PB-SMT).

In paper [4] Liang Tian, Derek F. Wong, Lidia S. Chao, Francisco Oliveira discusses the aspect of providing a formula to describe the relationship between word alignments, phrase table, and machine translation performance. The author of the papers also focuses on formulating such a relationship for estimating the size of extracted phrase pairs given one or more word alignment points.

In paper [5] Santos, André presents and describes the field of parallel corpora alignment. Its historical background and first development initiatives were described, and the alignment process was detailed step-by-step.

- **Parallel Text:** A parallel text is a set formed by a text and its translation (in which case it is called a bitext) or translations.
- **Parallel text alignment:** Parallel text alignment is the task of identifying correspondences between blocks or tokens in each half of a bitext.
- **Alignment Details:** The alignment of texts may be performed at several levels of granularity. Usually, when the objective is to achieve low-level alignment, like word alignment, higher-level alignment is performed first, which allows obtaining improved results.

2.3 FEASIBILITY STUDY

A feasibility study aims to objectively and rationally uncover the strengths and weakness of an existing business or proposed venture, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. A well-designed feasibility study should provide a historical background of the business or project, a description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements, and tax obligations.

During the feasibility analysis for this project, the following primary areas of interest are to be considered. Investigation and generating ideas about a new system does this.

2.3.1 TECHNICAL FEASIBILITY

Technical feasibility is a study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not. In our project the programming language that we are using python3 and also java8. We entirely work on Ubuntu version 14 because it is an open-source OS platform. For our project we use parsers mainly Stanford and Ace Parser, both are freely available. Most of the tools, packages, and resources we used are available for free to everyone, anyone can study, install and use them, and hence this project is technically feasible.

2.3.2 ECONOMICAL FEASIBILITY

Economic feasibility is the cost and logistical outlook for a business project or endeavor. Prior to embarking on a new venture, most businesses conduct an economic study, which is a study that analyses data to determine whether the cost of the prospective new venture will ultimately be profitable to the company. As all the

resources and tools we used in our system are open-source i.e. freely available to everyone, hence our project is Economically Feasible.

2.3.3 OPERATIONAL FEASIBILITY

As we used python3 for programming in our project, for the execution of the code which is written in the python language. The codes required can be easily modified by a person with decent python programming knowledge thus making it user-friendly. Thus, the system is operationally feasible as it very easy for the End users to operate it.

2.3.4 SCHEDULE FEASIBILITY

Time evaluation is the most important consideration in the development of the project. The schedule required for the development of this project is very important since more development time affects machine time, cost and causes a delay in the development of other systems. The modules were completed as per the schedule. It shows the estimated time to complete the project. This includes the schedules of each process in a project and the total project time. This can change if unexpected challenges occur. Schedule feasibility ensures that a project can be completed before the project or technology becomes obsolete or unnecessary. As our system is taking care of present requirements we need to ensure each time that each module of the system should be completed on time and will be more efficient than previous systems. Thus, schedule feasibility has been recovered.

3. PROPOSED SYSTEM

3.1 DRAWBACKS OF CURRENT SYSTEM AND NEED OF PROPOSED SYSTEM

Aligning words in translated texts is difficult for many reasons. Language differences make word-by-word translations very difficult in real translations. So manual word alignment is not an easy task and it can be very time-consuming. Today, there are effective techniques for automatic word alignment of parallel text. However, the quality of automatic word alignment tools and techniques currently present is not so good and accurate. Therefore, we are working on the possibility, to develop a software tool for automatic Word Phrase Alignment.

3.2 SYSTEM DESCRIPTION AND SRS

The Word Phrase Alignment tool/software we are trying to build is a research project at Anusaaraka lab, at IIIT Hyderabad. This tool/software if give satisfactory results after being completed, could be used as a part or module for their translation tool, which is the ultimate goal of the research.

Anusaaraka lab is concerned with the development of machine translation systems which in addition to the usual machine translation output also allows a user to understand the source language text in a pseudo target language. For example, a reader who knows Hindi (target language) would be able to read the English source text, in a pseudo-Hindi output after a small amount of training.

3.2.1 PURPOSE

The purpose is to build an automatic Word Phrase Alignment Tool with high accuracy and efficiency. The proposed system will take Parallel text as an input. The system will map the words/phrases from the source text to their corresponding words/phrase in the target text. The system will also extract translational information from the parallel text.

3.2.2 SCOPE OF PROJECT

The project is a part of research at the International Institute of Information Technology Hyderabad, under Anusaarka Lab for their open-source machine translation tool. We are working on the possibility of automating the task of alignment of words/phrases of the translated sentence in order to map these words/phrases with the words/phrases with the same meaning in the source sentence.

3.2.3 PROJECT PLANNING

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. Project planning is often used to organize different areas of a project, including project plans, workloads, and the management of teams and individuals.

Our project plan includes the study about Word alignment of bitext and concepts related to it such as parallel text, parallel text alignment, etc. And after that, we studied and reviewed a few alignment tools and techniques that were already present. Then we got the knowledge of how to group sentences into small phrases/chunks. After that, we started planning on how to write a code for grouping automation. After that, we wrote

two python scripts, one of which was for a minor task and the other was to access words and their POS-tags for the purpose of grouping. . Then we started analyzing and testing outputs of our programs and other tools and manually done work.

A Gantt chart is a horizontal bar chart developed as a production control tool in 1917 by Henry L. Gantt, an American engineer, and social scientist. A Gantt chart provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project. The following figure represents a Gantt chart of the system.

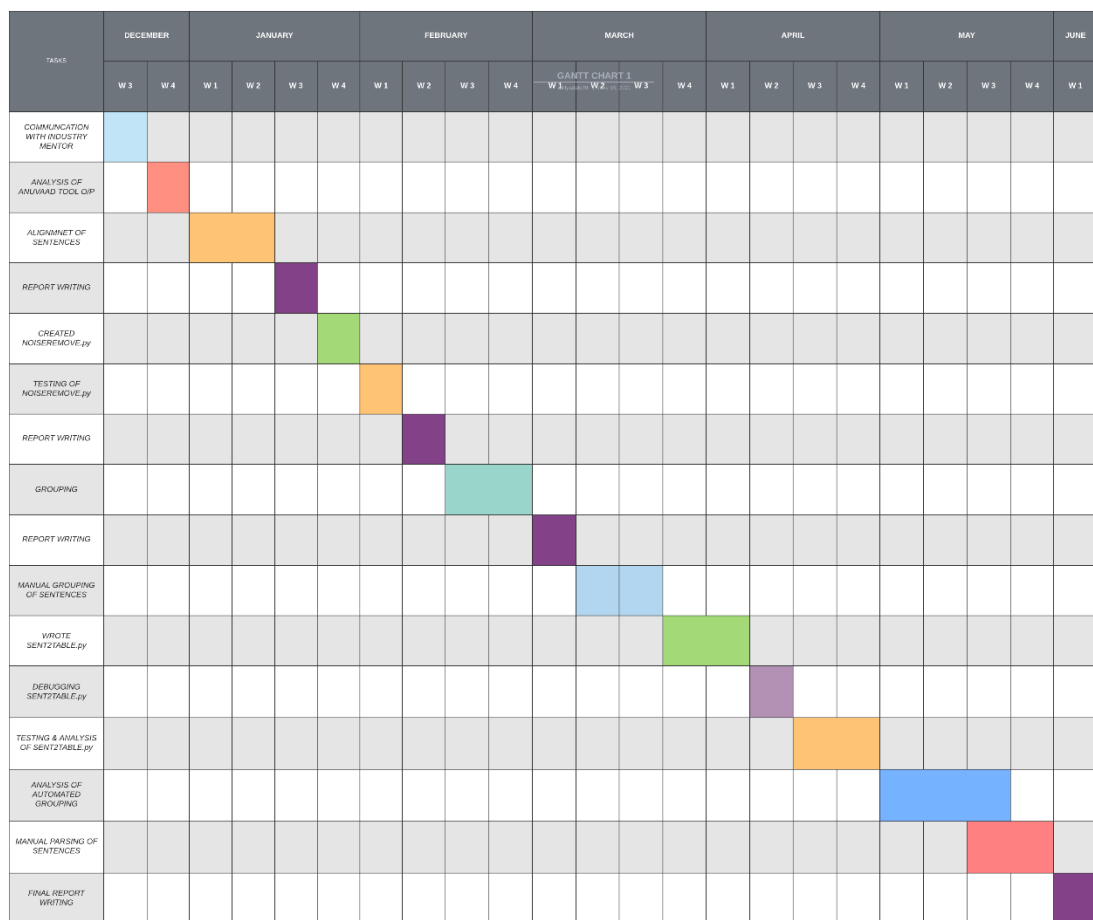


Fig: 3.2.3 Gantt chart

3.2.4 WORK-BREAKDOWN STRUCTURE

A work breakdown structure (WBS) in project management and systems engineering, is a deliverable-oriented breakdown of a project into smaller components. A work breakdown structure is a key project deliverable that organizes the team's work into manageable sections. A hierarchical decomposition of the total scope of work to be carried out by the project team to accomplish the project objectives and create the required deliverables.

TASK NO.	TASK	DURATION (DAYS)
1.	DISCOVERY	15
2.	PLANNING	25
3.	LITERATURE SURVEY	30
4.	SRS	15
5.	SYSTEM ANALYSIS	25
6.	SYSTEM DESIGN	10
7.	SYSTEM RELEASE	10
8.	DEPLOYMENT	10

3.3 SYSTEM ANALYSIS

The system analysis requirements gathering process is intensified and focused specifically on software. To understand the nature of the program to be built, the software engineer ("analyst") must understand the information domain for the software, as well as the required function, behavior, performance, and interface. Requirements for both the system and the software are documented and reviewed with the customer. System analysis is a process of collecting and interpreting facts, identifying the problems, and decomposition of a system into its components.

System analysis is conducted for studying a system or its parts to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

Here the work of the system is to generate languages which can be done by using python and clips. Hindi morph readme was not available, so to overcome this we have created the readme which contains the installation steps.

3.3.1 SYSTEM ARCHITECTURE

A system architecture is a conceptual model that defines the structure, behavior, and more views of a system. A system architecture can comprise system components that will work together to implement the overall system. Software architecture refers to the fundamental structures of a software system, the discipline of creating such structures, and the documentation of these structures. The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution are all described in the system diagram. One can think of system architecture as a set of representations of an existing (or future) system. Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements while

optimizing common quality attributes such as performance, security, and manageability.

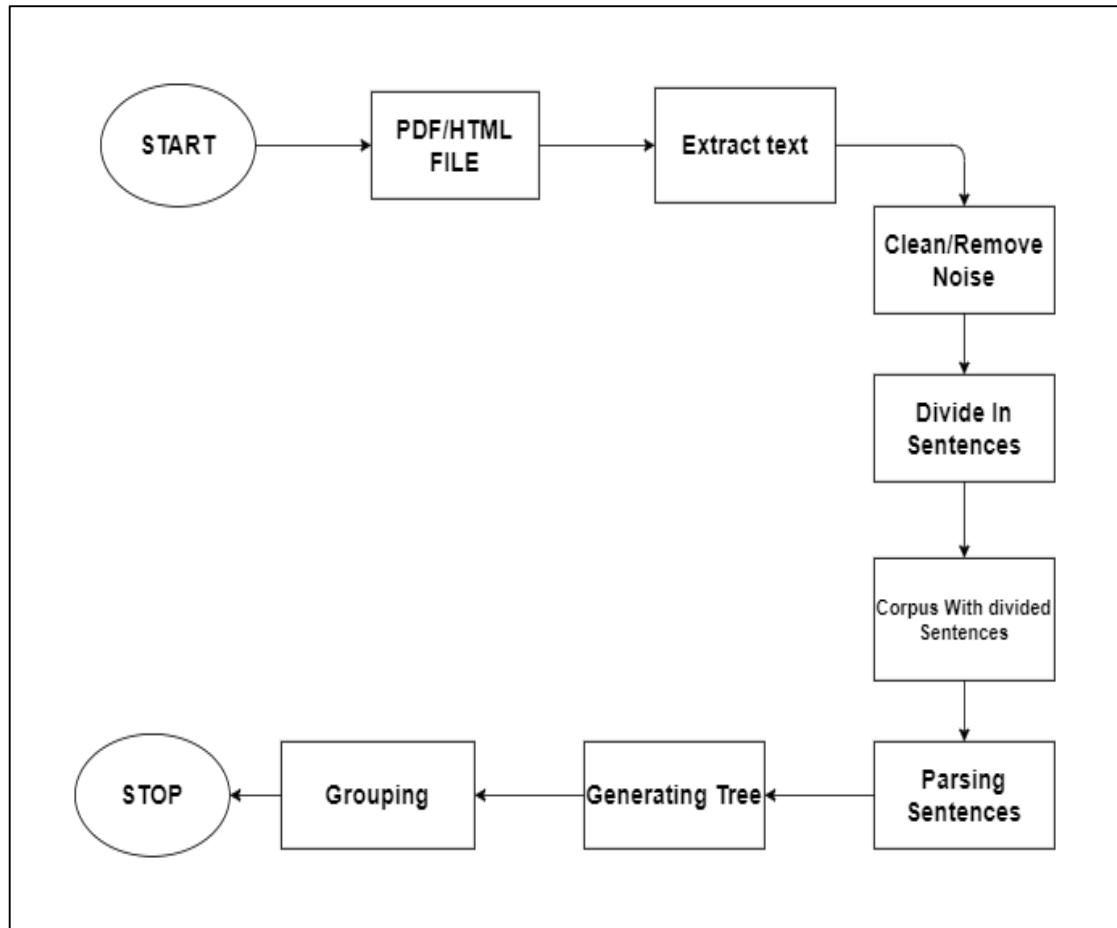


Fig: 3.3.1 System Architecture

In the System architecture shown above, first, the pdf/HTML file from which we want to extract sentences and remove the noises to create a clean corpus. In our project we used the book “Deep Learning” by MIT as a corpus, but it contained noise and the text that was irrelevant to us. So we use the python script "NioseRemove.py" that we wrote to remove the noise and create a clean corpus. After using NioseRemove.py we get a separate text file for a sentence. Then this sentence is parsed using Stanford Parser, through which we get its constituency parse and we generate the parse tree. Using this Parse tree we do a grouping of the sentence.

3.3.2 FLOWCHART OF SYSTEM

A flowchart is a type of diagram that represents an algorithm, workflow, or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting, or managing a process or program in various fields. The flowchart given below illustrates the flow of the website from the homepage to the various sections of the website and then ending at a common point.

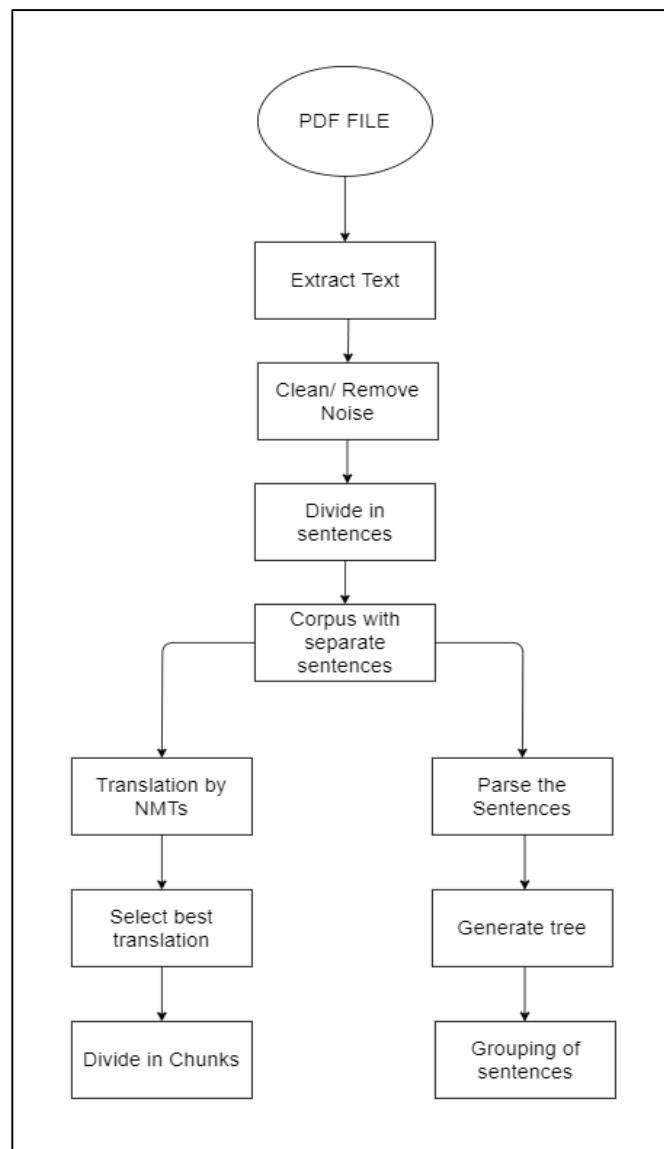


Fig: 3.3.2 Flow Chart

The above-shown flowchart shows how a corpus of separate sentences is created using python script "RemoveNoise.py" and how these sentences can be used for two different purposes.

3.3.3 DATA FLOW DIAGRAM

A data flow diagram is a graphical representation of the 'flow' of data through an information system, modeling its process aspects. The functional model consists of multiple DFD. A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. A level 0 data flow diagram (DFD), also known as a context diagram, shows a data system and emphasizes the way it interacts with external entities.

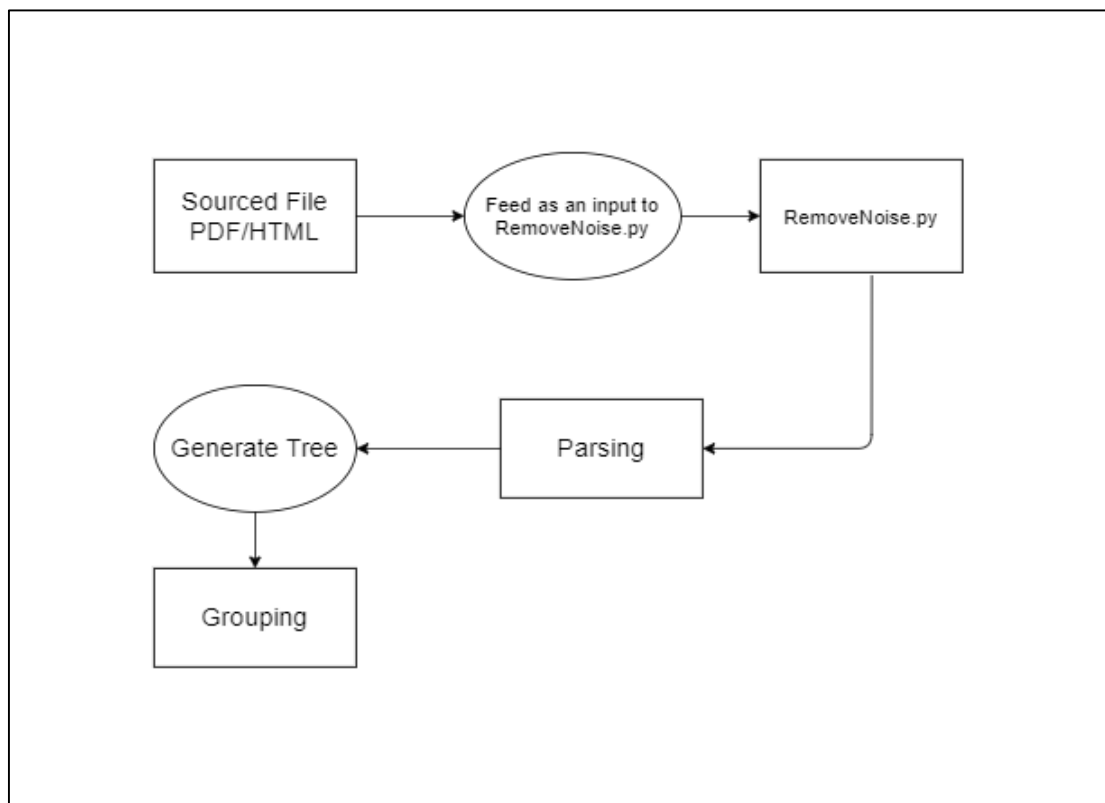


Fig: 3.3.3.1 DFD Level 0

In the level 0 DFD diagram, there will be an entity script that will provide the required output data which is then fed to the translator. The following figure shows the Level 0 DFD of the system.

The above shown DFD diagram Shows how the data flows from our source/ input file to the RemoveNoise.py python script and then it is feed to the Parser and how the data generated by parses as output is used to generate a parse tree.

A level 1 data flow diagram (DFD) is more detailed than a level 0 DFD but not as detailed as a level 2 DFD. It breaks down the main processes into subprocesses that can then be analyzed and improved on a more intimate level.

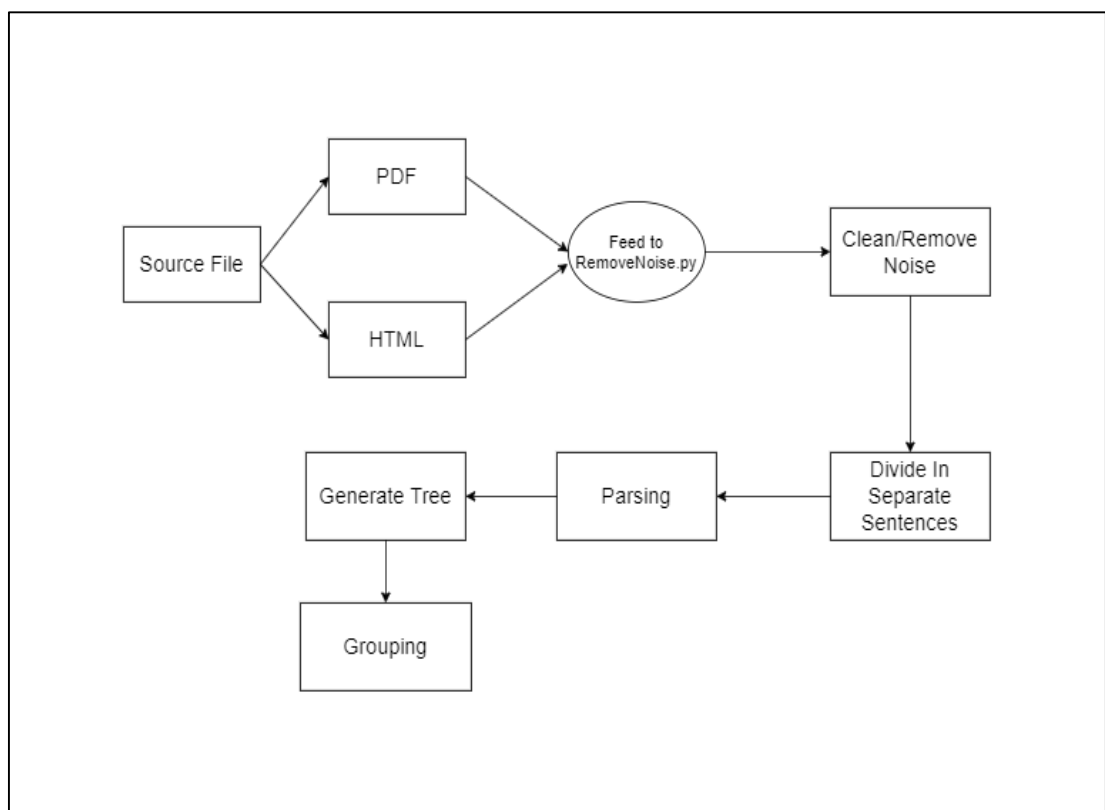


Fig: 3.3.3.2 DFD Level 1

The above-shown level 1 DFD diagram consists of entities developer's user will input source file containing natural language text and how we generate parse tree of a specific sentence from our source file.

3.3.4 USECASE DIAGRAM

A Use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

Here the developer will develop scripts to extract data from websites which is given to the system. The user will provide input source language and the system will process and give the output target language.

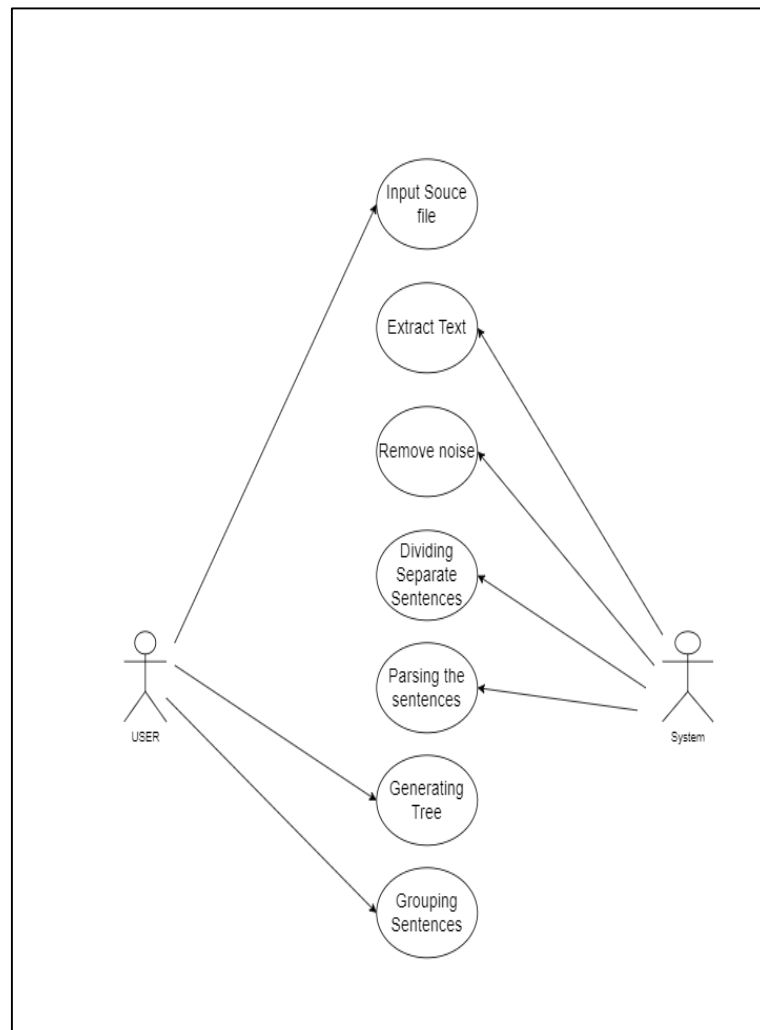


Fig: 3.3.4 Use Case Diagram

Fig 3.3.4, shows how the user will provide input source file, and the system will extract, clean, and process the text from the source file and give the output as clean separate sentences for grouping purposes.

3.3.5 HARDWARE REQUIREMENTS

- Processor: Intel i3 or higher
- RAM: 4 GB or higher
- Disk Space: 1 GB of free disk space

3.3.6 SOFTWARE REQUIREMENTS

- Windows XP or later, Ubuntu
- Python 3.6 or above
- Java 8 or above

3.4 SYSTEM DESIGN

Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. The logical system design arrived because systems analysis is converted into physical system design. Normally, the design proceeds in two stages: (1) Preliminary or General Design, (2) Structured or Detailed Design.

In the preliminary or general design, the features of the new system are specified. The costs of implementing these features and the benefits to be derived are estimated. If the project is still considered to be feasible, we move to the detailed design stage.

The system design is of three types which are architectural design, logical design, and physical design. The architectural design of a system emphasizes on the design of the systems architecture which describes the structure, behavior, and more views of that

system and analysis. The logical design of a system pertains to an abstract representation of the data flows, inputs, and outputs of the system.

3.4.1 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects.

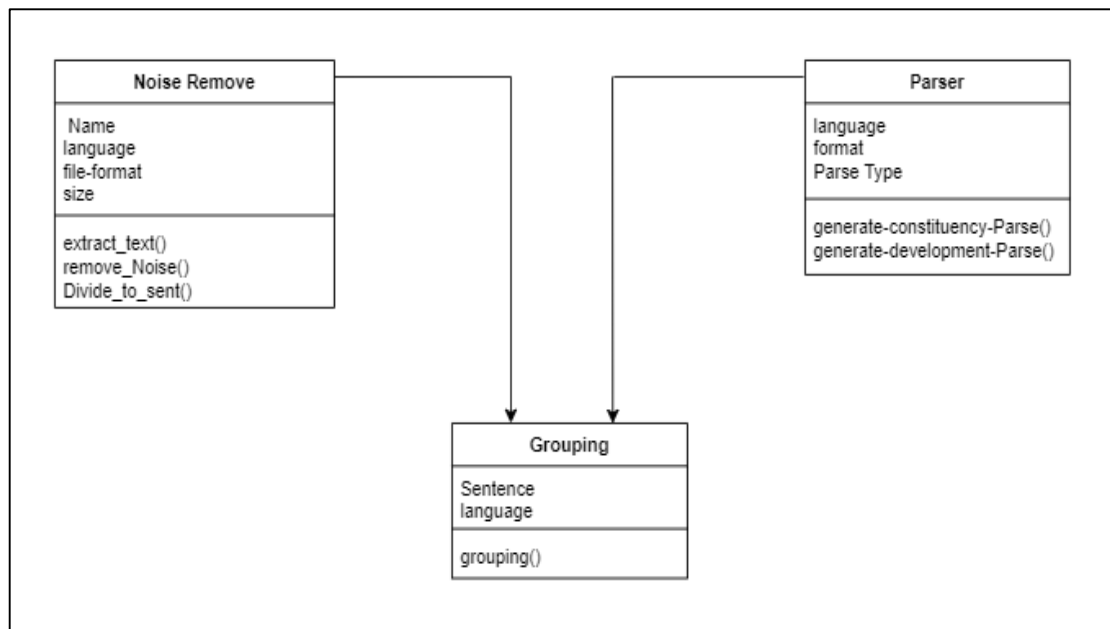


Fig: 3.4.1 Class Diagram

Fig. 3.4.1, Shows the attributes and behaviors of the classes present in our system.

3.4.2 ACTIVITY DIAGRAM

The activity diagram is another important diagram in UML to describe the dynamic aspects of the system. The activity diagram is a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc. The basic purposes of activity diagrams are similar to the other four diagrams. It captures the dynamic behavior of the system. The other four diagrams are used to show the message flow from one object to another but the activity diagram is used to show message flow from one activity to another.

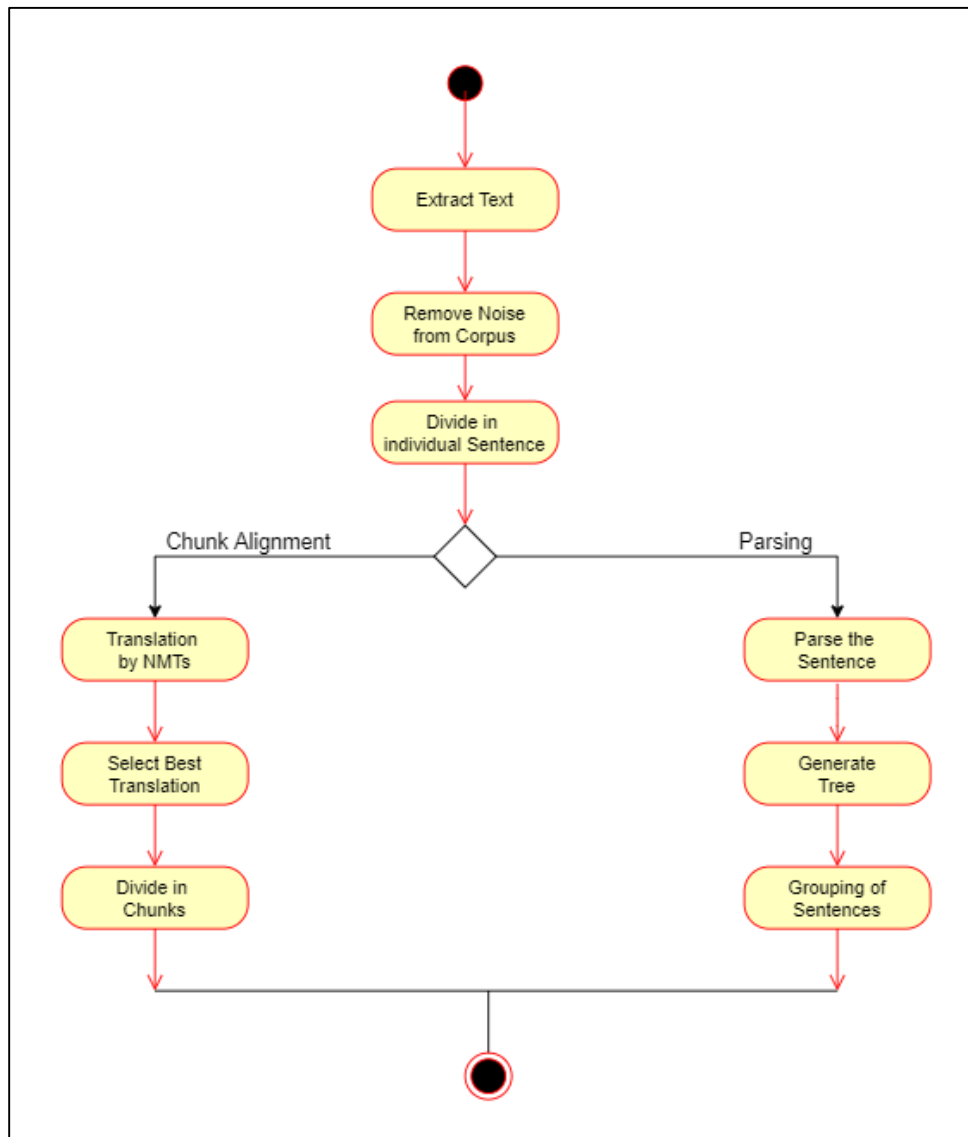


Fig: 3.4.2 Activity Diagram

Fig. 3.4.2, shows the flow control of our system. How the control starts to flow from the input and travels through NioseRemve.py to Stanford parser, etc.

3.4.3 COMPONENT DIAGRAM

Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering.

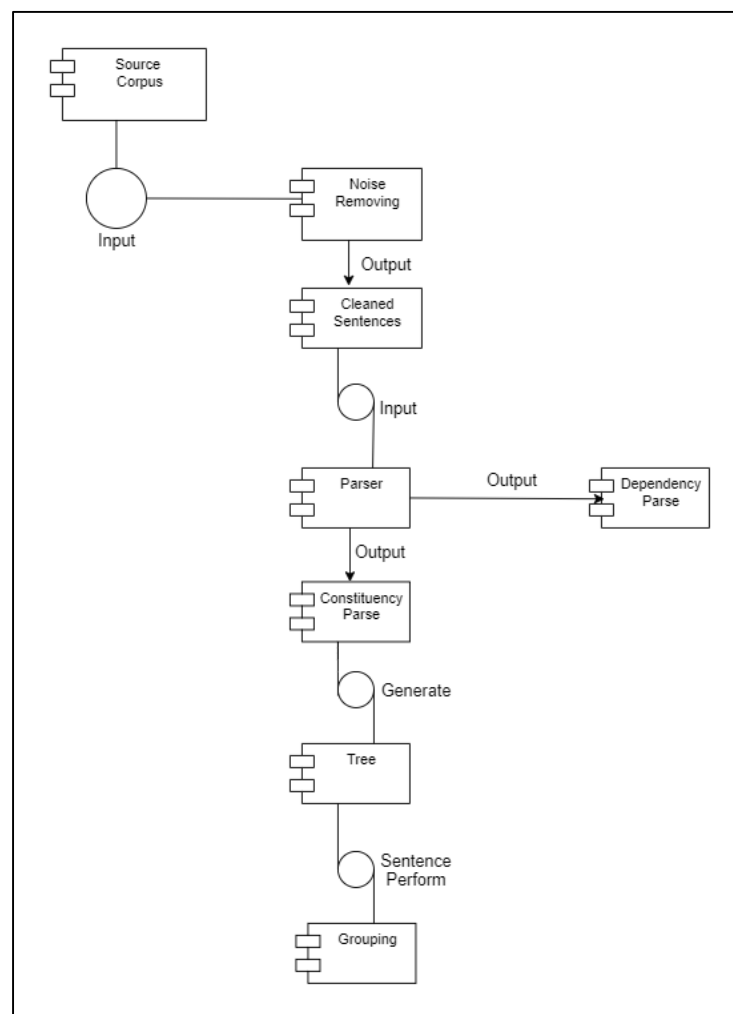


Fig. 3.4.3 Component Diagram

Fig. 3.4.3, shows the components present in our system such as Parser, RemoveNoise python program, etc. It also shows how these components are related.

3.4.4 DEPLOYMENT DIAGRAM

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system.

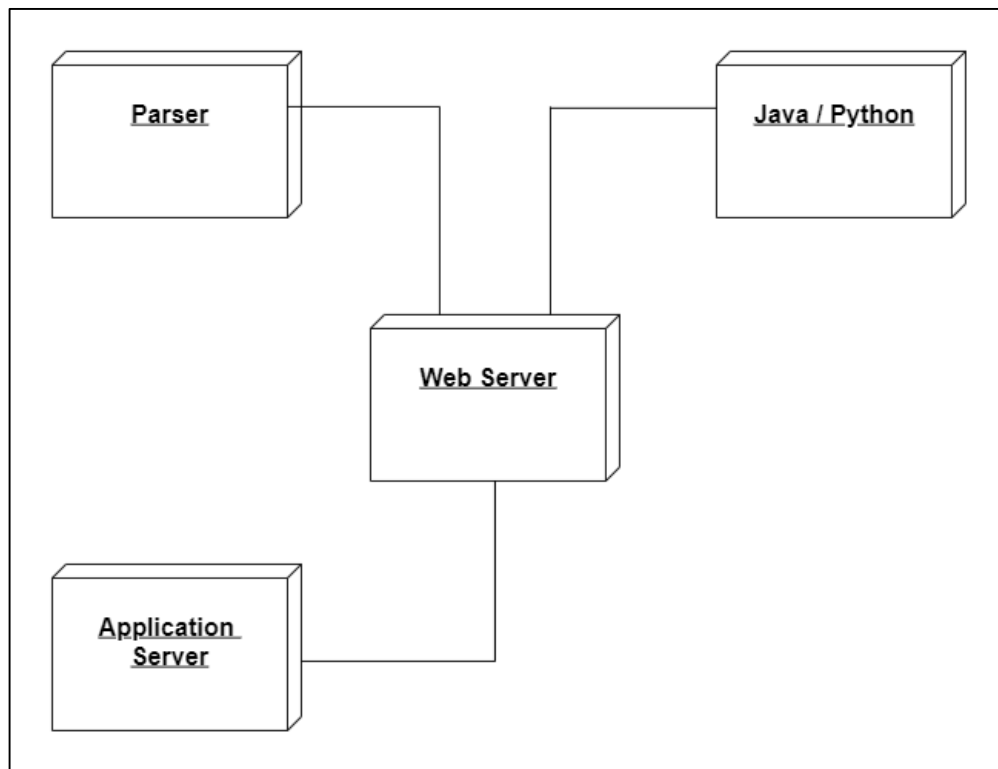


Fig: 3.4.4 Deployment Diagram.

Fig. 3.4.4, shows how our system once completed will be deployed for real-world use.

4. IMPLEMENTATION AND CODING

4.1 SYSTEM RELEASE PLAN

Release management has been a core process of software development for decades. The purpose of release management processes is to coordinate the development, operations, deployment of software while ensuring alignment with business priorities.

SPRINTS	TASK NO.	DETAILS	EXPECTED RELEASE DATE	REMARK & SIGNED BY PROJECT GUIDE/INDUSTRY MENTOR
SPRINT 0	1.	Software setup and installation	13 th December 2020	
SPRINT 1	2.	Understanding the need of the project.	20 th December 2020	
	3.	Learning the prerequisite of the project (python, java, parsing, etc.)		
SPRINT 2	4.	Studying about Word Phrase Alignment	03 rd January 2021	
	5.	Studying about existing tools for Word Phrase Alignment		
	6.	Analyzing Project Requirement		
SPRINT 3	7.	Analyzed outputs of Anuvaad Tool to	24 th January 2021	

		identify any shortcomings		
	8.	Manually aligning sentences in chunk alignment.		
	9.	Creating an algorithm for alignment		
SPRINT 4	10.	Created a program for cleaning and extraction of text.	28 th February 2021	
SPRINT 5	11.	Learn about the grouping of sentences.	25 th March 2021	
	12.	Performed Manual grouping sentences		
	13.	Created recursive program based on constituency		
SPRINT 6	14.	Compared o/p of manual and automated grouping	29 th April 2021	
	15.	Understand how to parse is done		

4.2 IMPLEMENTATION METHODOLOGY

4.2.1 NOISE REMOVE

This program removes unwanted noise from the corpus and extracts text.

1. The input of this program is a corpus, taken from a book.

```
CHAPTER 1. INTRODUCTION
so has the complexity of the tasks that they can solve. Goodfellow et al. (2014d)
showed that neural networks could learn to output an entire sequence of characters
transcribed from an image, rather than just identifying a single object. Previously,
it was widely believed that this kind of learning required labeling of the individual
elements of the sequence (Gülçehre and Bengio, 2013). Recurrent neural networks,
such as the LSTM sequence model mentioned above, are now used to model
relationships between sequences and other sequences rather than just fixed inputs.
This sequence-to-sequence learning seems to be on the cusp of revolutionizing
another application: machine translation (Sutskever et al., 2014; Bahdanau et al.,
2015).
This trend of increasing complexity has been pushed to its logical conclusion
with the introduction of neural Turing machines (Graves et al., 2014) that learn
to read from memory cells and write arbitrary content to memory cells. Such
neural networks can learn simple programs from examples of desired behavior. For
example, they can learn to sort lists of numbers given examples of scrambled and
sorted sequences. This self-programming technology is in its infancy, but in the
future it could in principle be applied to nearly any task.
Another crowning achievement of deep learning is its extension to the domain of
reinforcement learning
. In the context of reinforcement learning, an autonomous
agent must learn to perform a task by trial and error, without any guidance from
the human operator. DeepMind demonstrated that a reinforcement learning system
based on deep learning is capable of learning to play Atari video games, reaching
human-level performance on many tasks (Mnih et al., 2015). Deep learning has
also significantly improved the performance of reinforcement learning for robotics
(Finn et al., 2015).
Many of these applications of deep learning are highly profitable. Deep learning
is now used by many top technology companies, including Google, Microsoft,
Facebook, IBM, Baidu, Apple, Adobe, Netflix, NVIDIA, and NEC.
Advances in deep learning have also depended heavily on advances in software
infrastructure. Software libraries such as Theano (Bergstra et al., 2010; Bastien
et al., 2012), PyLearn2 (Goodfellow et al., 2013c), Torch (Collobert et al., 2011b),
DistBelief (Dean et al., 2012), Caffe (Jia, 2013), MXNet (Chen et al., 2015), and
TensorFlow (Abadi et al., 2015) have all supported important research projects or
commercial products.
Deep learning has also made contributions to other sciences. Modern convolu-
tional networks for object recognition provide a model of visual processing that
neuroscientists can study (DiCarlo, 2013). Deep learning also provides useful tools
for processing massive amounts of data and making useful predictions in scientific
25
CHAPTER 1. INTRODUCTION
fields. It has been successfully used to predict how molecules will interact in order
to help pharmaceutical companies design new drugs (Dahl et al., 2014), to search
for subatomic particles (Baldi et al., 2014), and to automatically parse microscope
images used to construct a 3-D map of the human brain (Knowles-Barley et al.,
2014). We expect deep learning to appear in more and more scientific fields in the
future.
In summary, deep learning is an approach to machine learning that has drawn
heavily on our knowledge of the human brain, statistics and applied math as it
developed over the past several decades. In recent years, deep learning has seen
```

Fig: 4.2.1.1 Input of program Noise Remove

2. The output of the program returns a text file with a cleaned corpus i.e. unwanted noises and irrelevant data have been removed (such as Chapter Name, Chapter No., Page No., etc.).

so has the complexity of the tasks that they can solve. Goodfellow et al. (2014d) showed that neural networks could learn to output an entire sequence of characters transcribed from an image, rather than just identifying a single object. Previously, it was widely believed that this kind of learning required labeling of the individual elements of the sequence (Gülçehre and Bengio, 2013). Recurrent neural networks, such as the LSTM sequence model mentioned above, are now used to model relationships between sequences and other sequences rather than just fixed inputs. This sequence-to-sequence learning seems to be on the cusp of revolutionizing another application: machine translation (Sutskever et al., 2014; Bahdanau et al., 2015).

This trend of increasing complexity has been pushed to its logical conclusion with the introduction of neural Turing machines (Graves et al., 2014) that learn to read from memory cells and write arbitrary content to memory cells. Such neural networks can learn simple programs from examples of desired behavior. For example, they can learn to sort lists of numbers given examples of scrambled and sorted sequences. This self-programming technology is in its infancy, but in the future it could in principle be applied to nearly any task.

Another crowning achievement of deep learning is its extension to the domain of reinforcement learning.

. In the context of reinforcement learning, an autonomous agent must learn to perform a task by trial and error, without any guidance from the human operator. DeepMind demonstrated that a reinforcement learning system based on deep learning is capable of learning to play Atari video games, reaching human-level performance on many tasks (Mnih et al., 2015). Deep learning has also significantly improved the performance of reinforcement learning for robotics (Finn et al., 2015).

Many of these applications of deep learning are highly profitable. Deep learning is now used by many top technology companies, including Google, Microsoft, Facebook, IBM, Baidu, Apple, Adobe, Netflix, NVIDIA, and NEC.

Advances in deep learning have also depended heavily on advances in software infrastructure. Software libraries such as Theano (Bergstra et al., 2010; Bastien et al., 2012), PyLearn2 (Goodfellow et al., 2013c), Torch (Collobert et al., 2011b), DistBelief (Dean et al., 2012), Caffe (Jia, 2013), MXNet (Chen et al., 2015), and TensorFlow (Abadi et al., 2015) have all supported important research projects or commercial products.

Deep learning has also made contributions to other sciences. Modern convolutional networks for object recognition provide a model of visual processing that neuroscientists can study (DiCarlo, 2013). Deep learning also provides useful tools for processing massive amounts of data and making useful predictions in scientific fields. It has been successfully used to predict how molecules will interact in order to help pharmaceutical companies design new drugs (Dahl et al., 2014), to search for subatomic particles (Baldi et al., 2014), and to automatically parse microscope images used to construct a 3-D map of the human brain (Knowles-Barley et al., 2014). We expect deep learning to appear in more and more scientific fields in the future.

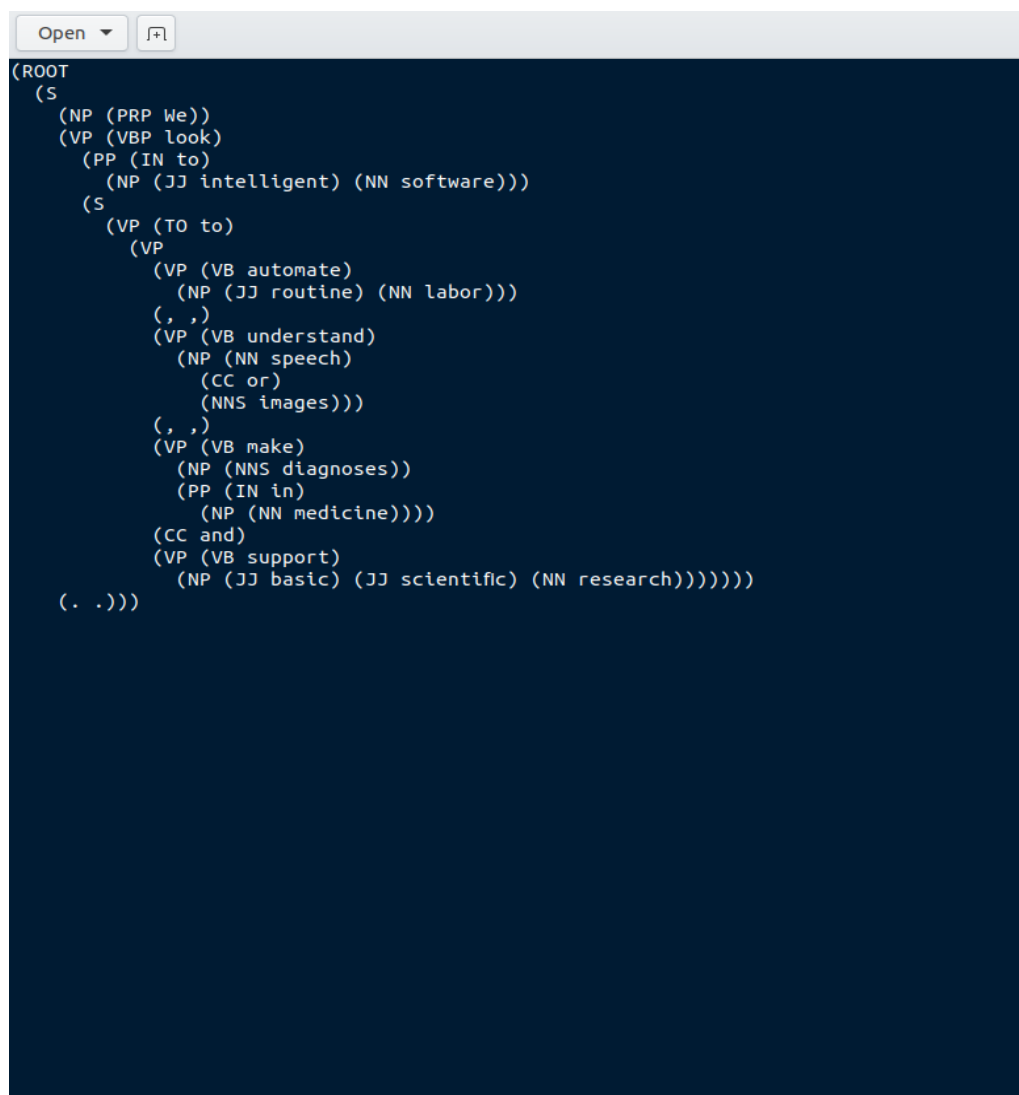
In summary, deep learning is an approach to machine learning that has drawn heavily on our knowledge of the human brain, statistics and applied math as it developed over the past several decades. In recent years, deep learning has seen

Fig: 4.2.1.2 Output of program Noise Remove

4.2.2 PROGRAM SENT2TABLE

This program extracts useful information from constituency parse of a sentence which is useful for grouping.

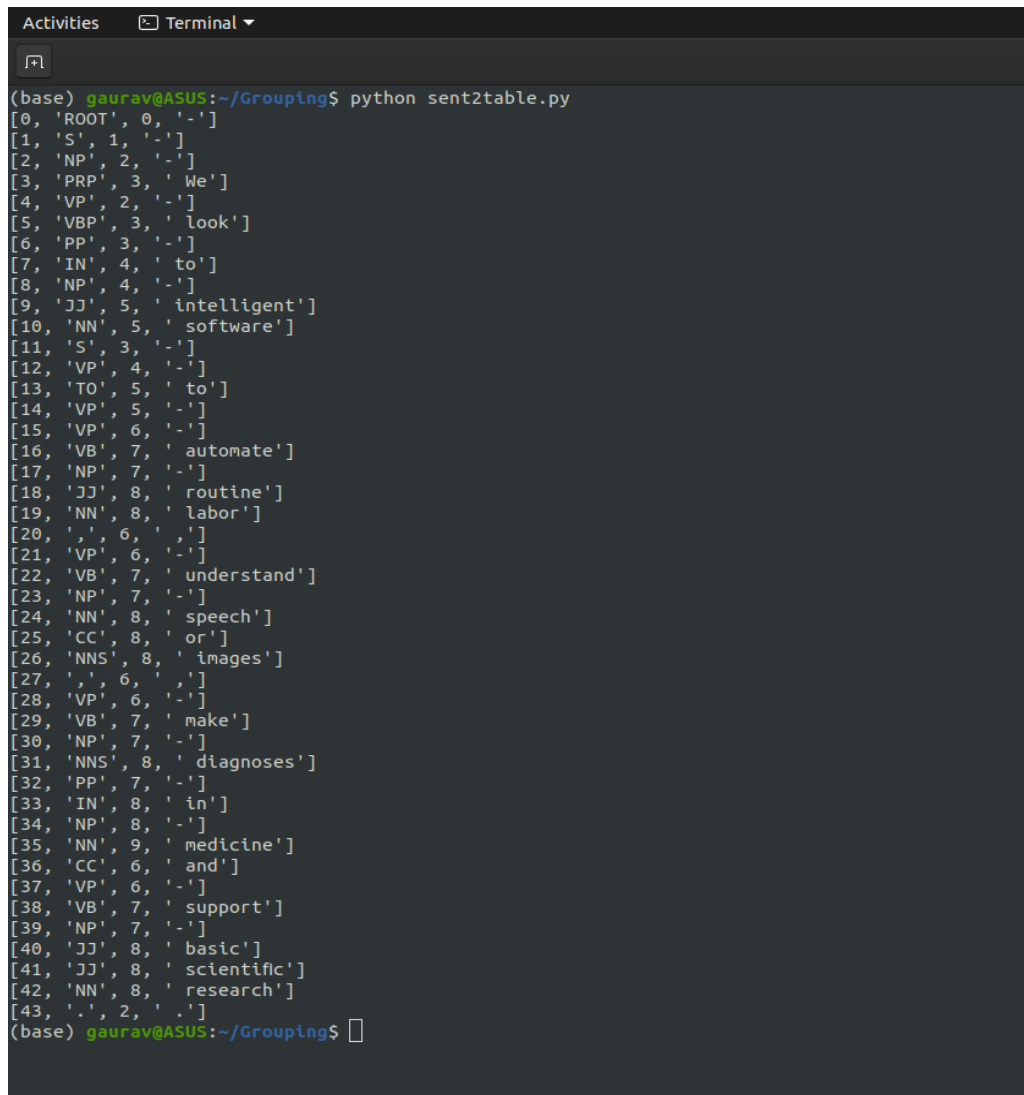
1. The input contains the constituency parse of a sentence, generated using a parser (like Stanford, ACE, etc.).



```
(ROOT
  (S
    (NP (PRP We))
    (VP (VBP look)
      (PP (IN to)
        (NP (JJ intelligent) (NN software)))
      (S
        (VP (TO to)
          (VP (VP (VB automate)
              (NP (JJ routine) (NN labor)))
            (, ,)
            (VP (VB understand)
              (NP (NN speech)
                (CC or)
                (NNS images)))
            (, ,)
            (VP (VB make)
              (NP (NNS diagnoses)
                (PP (IN in)
                  (NP (NN medicine))))
              (CC and)
              (VP (VB support)
                (NP (JJ basic) (JJ scientific) (NN research))))))
          (NP (JJ basic) (JJ scientific) (NN research))))))
    (. .)))
```

Fig: 4.2.2.1 Input of program Sent2table

2. The output consists of a table that contains data about each node of the Parse Tree.
- The table consists of the following information about each node i.e. ID, level, POS Tag and Word.



```
(base) gaurav@ASUS:~/Grouping$ python sent2table.py
[0, 'ROOT', 0, '-']
[1, 'S', 1, '-']
[2, 'NP', 2, '-']
[3, 'PRP', 3, 'We']
[4, 'VP', 2, '-']
[5, 'VBP', 3, 'look']
[6, 'PP', 3, '-']
[7, 'IN', 4, 'to']
[8, 'NP', 4, '-']
[9, 'JJ', 5, 'intelligent']
[10, 'NN', 5, 'software']
[11, 'S', 3, '-']
[12, 'VP', 4, '-']
[13, 'TO', 5, 'to']
[14, 'VP', 5, '-']
[15, 'VP', 6, '-']
[16, 'VB', 7, 'automate']
[17, 'NP', 7, '-']
[18, 'JJ', 8, 'routine']
[19, 'NN', 8, 'labor']
[20, ',', 6, ',']
[21, 'VP', 6, '-']
[22, 'VB', 7, 'understand']
[23, 'NP', 7, '-']
[24, 'NN', 8, 'speech']
[25, 'CC', 8, 'or']
[26, 'NNS', 8, 'images']
[27, ',', 6, ',']
[28, 'VP', 6, '-']
[29, 'VB', 7, 'make']
[30, 'NP', 7, '-']
[31, 'NNS', 8, 'diagnoses']
[32, 'PP', 7, '-']
[33, 'IN', 8, 'in']
[34, 'NP', 8, '-']
[35, 'NN', 9, 'medicine']
[36, 'CC', 6, 'and']
[37, 'VP', 6, '-']
[38, 'VB', 7, 'support']
[39, 'NP', 7, '-']
[40, 'JJ', 8, 'basic']
[41, 'JJ', 8, 'scientific']
[42, 'NN', 8, 'research']
[43, '.', 2, '.']
(base) gaurav@ASUS:~/Grouping$
```

Fig: 4.2.2.2 Output of program Sent2table

4.2.3 PARSER MODIFICATION

This python script performs automated grouping by using the constituency parse of the sentence.

1. This script takes a text file containing a sentence as an input.

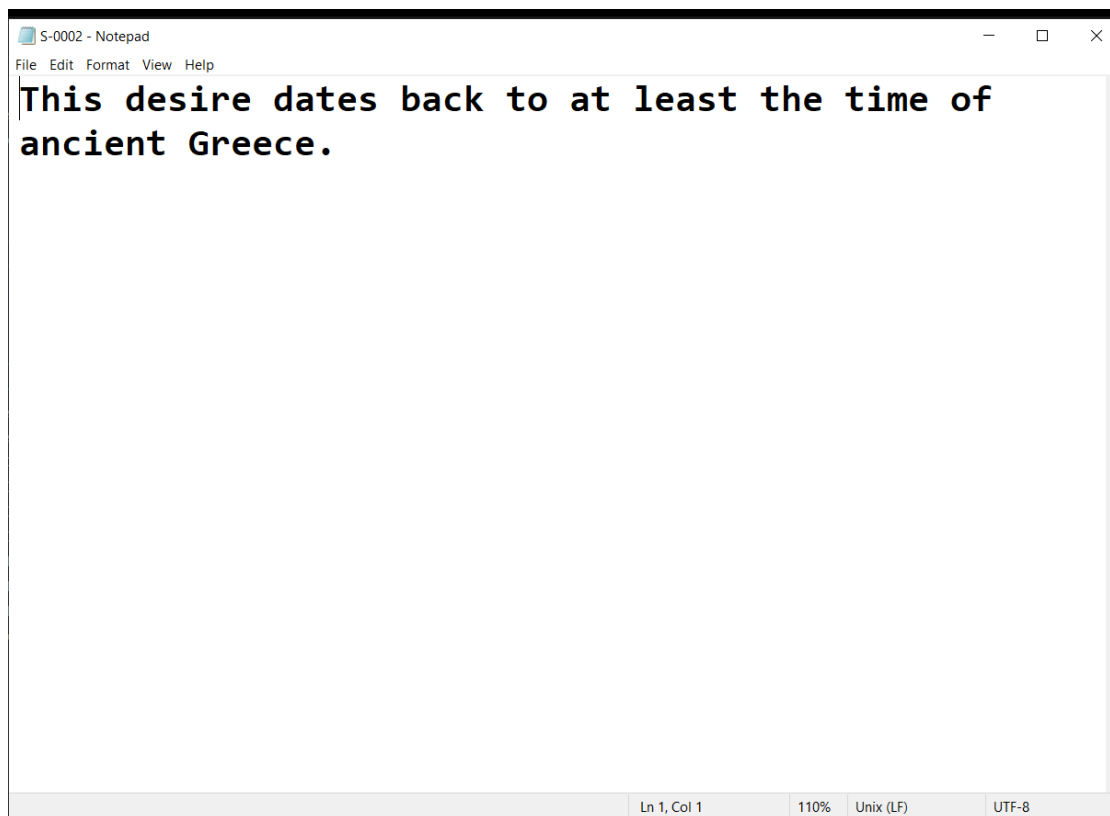
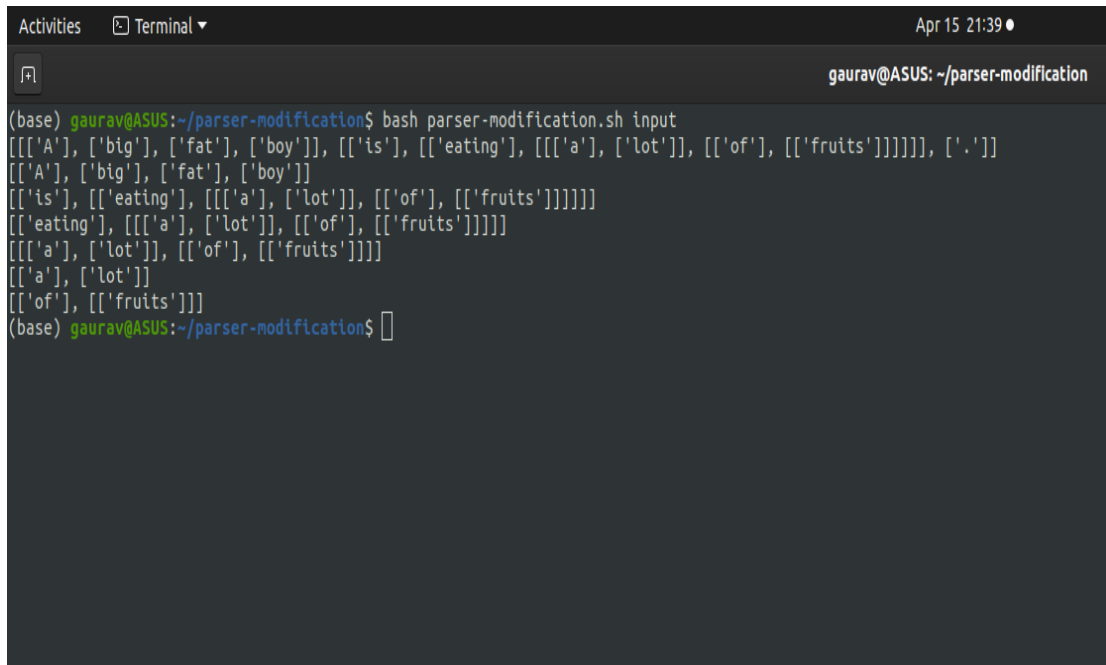


Fig: 4.2.3.1 Input of program Parser Modification

2. The output of this program returns the grouping of the sentence based on its constituency parse.



```
(base) gaurav@ASUS:~/parser-modification$ bash parser-modification.sh input
[[['A', ['big', ['fat', ['boy']], ['is', ['eating', [['a', ['lot']], ['of', [['fruits']]]]]], ['.']]
[['A', ['big', ['fat', ['boy']]
[['is', ['eating', [['a', ['lot']], ['of', [['fruits']]]]]]]
[['eating', [['a', ['lot']], ['of', [['fruits']]]]]
[['a', ['lot']], ['of', [['fruits']]]
[['a', ['lot']]
[['of', [['fruits']]]
(base) gaurav@ASUS:~/parser-modification$
```

Fig: 4.2.3.2 Output of program Parser Modification

4.3 CODE AND SCREENSHOTS

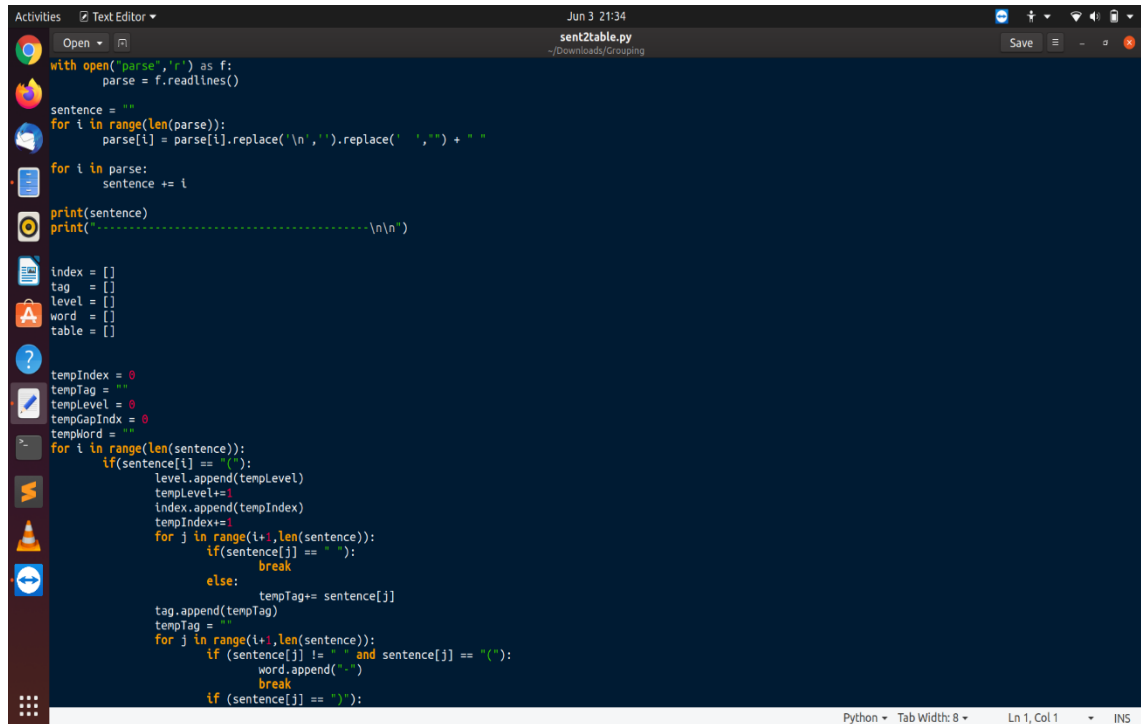


Fig: 4.3.1 Snapshot of Program Sent2table.py

Program Sent2table.py:

```
with open("parse", 'r') as f:  
    parse = f.readlines()
```

```
sentence = ""  
for i in range(len(parse)):  
    parse[i] = parse[i].replace("\n", "").replace(' ', "") + " "
```

```
for i in parse:  
    sentence += i
```

```
print(sentence)  
print("-----\n\n")
```

```
index = []  
tag = []  
level = []  
word = []  
table = []
```



```

tempIndex = 0
tempTag = ""
tempLevel = 0
tempGapIndx = 0
tempWord = ""
for i in range(len(sentence)):
    if(sentence[i] == "("):
        level.append(tempLevel)
        tempLevel+=1
        index.append(tempIndex)
        tempIndex+=1
        for j in range(i+1,len(sentence)):
            if(sentence[j] == " "):
                break
            else:
                tempTag+= sentence[j]
                tag.append(tempTag)
                tempTag = ""
                for j in range(i+1,len(sentence)):
                    if (sentence[j] != " " and sentence[j] == "("):
                        word.append("-")
                        break
                    if (sentence[j] == ")"):
                        for x in range(i+1,len(sentence)):
                            if (sentence[x] == " "):
                                tempGapIndx = x
                                break
                        for x in range(tempGapIndx,len(sentence)):
                            if (sentence[x] == ")"):
                                tempGapIndx = 0
                                break
                        else:
                            tempWord += sentence[x]
                            word.append(tempWord)
                            tempWord = ""
                            break

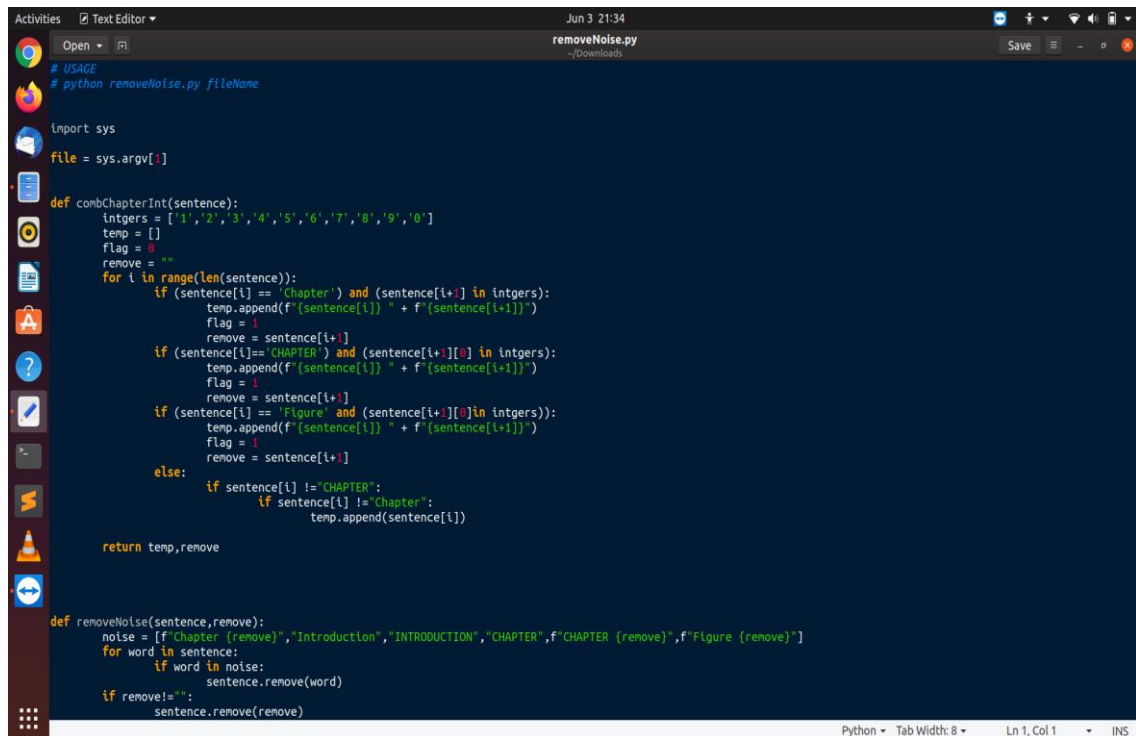
    if(sentence[i] == ")"):
        tempLevel -= 1

tempTable = []

for i in range(len(index)):
    tempTable.append(index[i])
    tempTable.append(tag[i])
    tempTable.append(level[i])
    tempTable.append(word[i])
    table.append(tempTable)
    tempTable = []

```

for i in table:
print(i)



```
# USAGE
# python removeNoise.py fileName

import sys
file = sys.argv[1]

def combChapterInt(sentence):
    integers = ['1','2','3','4','5','6','7','8','9','0']
    temp = []
    flag = 0
    remove = ""
    for i in range(len(sentence)):
        if (sentence[i] == 'Chapter') and (sentence[i+1] in integers):
            temp.append(f'{sentence[i]} ' + f'{sentence[i+1]}')
            flag = 1
            remove = sentence[i+1]
        if (sentence[i]=='CHAPTER') and (sentence[i+1][0] in integers):
            temp.append(f'{sentence[i]} ' + f'{sentence[i+1]}')
            flag = 1
            remove = sentence[i+1]
        if (sentence[i] == 'Figure' and (sentence[i+1][0] in integers)):
            temp.append(f'{sentence[i]} ' + f'{sentence[i+1]}')
            flag = 1
            remove = sentence[i+1]
        else:
            if sentence[i] != 'CHAPTER':
                if sentence[i] != 'Chapter':
                    temp.append(sentence[i])
    return temp,remove

def removeNoise(sentence,remove):
    noise = [f'Chapter {remove}',"Introduction","INTRODUCTION","CHAPTER",f'CHAPTER {remove}',f'Figure {remove}']
    for word in sentence:
        if word in noise:
            sentence.remove(word)
    if remove!="":
        sentence.remove(remove)
```

Fig: 4.3.2 Snapshot of Program removeNoise.py

Program removeNoise.py:

```
# USAGE
# python removeNoise.py fileName

import sys
file = sys.argv[1]

def combChapterInt(sentence):
    integers = ['1','2','3','4','5','6','7','8','9','0']
    temp = []
    flag = 0
    remove = ""
    for i in range(len(sentence)):
        if (sentence[i] == 'Chapter') and (sentence[i+1] in integers):
            temp.append(f'{sentence[i]} ' + f'{sentence[i+1]}')
            flag = 1
            remove = sentence[i+1]
        if (sentence[i]=='CHAPTER') and (sentence[i+1][0] in integers):
            temp.append(f'{sentence[i]} ' + f'{sentence[i+1]}')
```

```

flag = 1
remove = sentence[i+1]
if (sentence[i] == 'Figure' and (sentence[i+1][0] in intgers)):
temp.append(f"{sentence[i]} " + f"{sentence[i+1]}")
flag = 1
remove = sentence[i+1]
else:
if sentence[i] != "CHAPTER":
if sentence[i] != "Chapter":
temp.append(sentence[i])

return temp,remove

def removeNoise(sentence,remove):
noise = [f"Chapter
{remove}", "Introduction", "INTRODUCTION", "CHAPTER", f"CHAPTER
{remove}", f"Figure {remove}"]
for word in sentence:
if word in noise:
sentence.remove(word)
if remove!="":
sentence.remove(remove)
return sentence

temp = []
count = 0

with open(file,"r") as read:
sentence = read.readlines()

for i in range(len(sentence)):
sentence[i] = sentence[i].split()

for i in range(len(sentence)):
sent,remove = combChapterInt(sentence[i])
sentenceToWrite = removeNoise(sent,remove)
temp.append(sentenceToWrite)
count +=1

final = ""
for i in range(count):
for j in range(len(temp[i])):
final += temp[i][j] + " "
final += '\n'

with open(f"{sys.argv[1]}_noiseRemoved.txt","w") as write:
write.writelines(final)

```

5. TESTING

5.1 INTRODUCTION

Testing is the set of activities that can be planned and conducted systematically testing requires that the developer discard preconceived notions of the correctness of the software just developed and overcome a conflict of interest that occurs when errors are encountered. Testing also provides the main objective of our project and understands the risk of implementation. Testing is a process of technical investigation, performed on behalf of stakeholders that is intended to reveal quantity-related information about the product concerning the context in which it is intended to operate. Testing is the process of executing a program or an application with the intent of finding an error or bugs. Testing can be stated as the process of validating and verifying that a software program/application/product.

5.2 FUNCTIONALITY TESTING

Functional testing is a quality assurance (QA) process and a type of black-box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (unlike white box testing. Functional testing is conducted to evaluate the compliance of a system or component with specified functional requirements. Functional testing usually describes what the system does.

```
Activities Terminal
s(base) gaurav@ASUS:~/Desktop/Demo/sent2Table$ python sent2table.py
[0, 'ROOT', 0, '-']
[1, 'S', 1, '-']
[2, 'NP', 2, '-']
[3, 'PRP', 3, 'We']
[4, 'VP', 2, '-']
[5, 'VBP', 3, 'look']
[6, 'PP', 3, '-']
[7, 'IN', 4, 'to']
[8, 'NP', 4, '-']
[9, 'JJ', 5, 'intelligent']
[10, 'NN', 5, 'software']
[11, 'S', 3, '-']
[12, 'VP', 4, '-']
[13, 'TO', 5, 'to']
[14, 'VP', 5, '-']
[15, 'VP', 6, '-']
[16, 'VB', 7, 'automate']
[17, 'NP', 7, '-']
[18, 'JJ', 8, 'routine']
[19, 'NN', 8, 'labor']
[20, ',', 6, ',']
[21, 'VP', 6, '-']
[22, 'VB', 7, 'understand']
[23, 'NP', 7, '-']
[24, 'NN', 8, 'speech']
[25, 'CC', 8, 'or']
[26, 'NNS', 8, 'images']
[27, ',', 6, ',']
[28, 'VP', 6, '-']
[29, 'VB', 7, 'make']
[30, 'NP', 7, '-']
[31, 'NNS', 8, 'diagnoses']
[32, 'PP', 7, '-']
[33, 'IN', 8, 'in']
[34, 'NP', 8, '-']
[35, 'NN', 9, 'medicine']
[36, 'CC', 6, 'and']
[37, 'VP', 6, '-']
[38, 'VB', 7, 'support']
[39, 'NP', 7, '-']
[40, 'JJ', 8, 'basic']
[41, 'JJ', 8, 'scientific']
[42, 'NN', 8, 'research']
[43, '.', 2, '.']
(base) gaurav@ASUS:~/Desktop/Demo/sent2Table$
```

Fig: 5.2 Functionality testing of sent2table.py

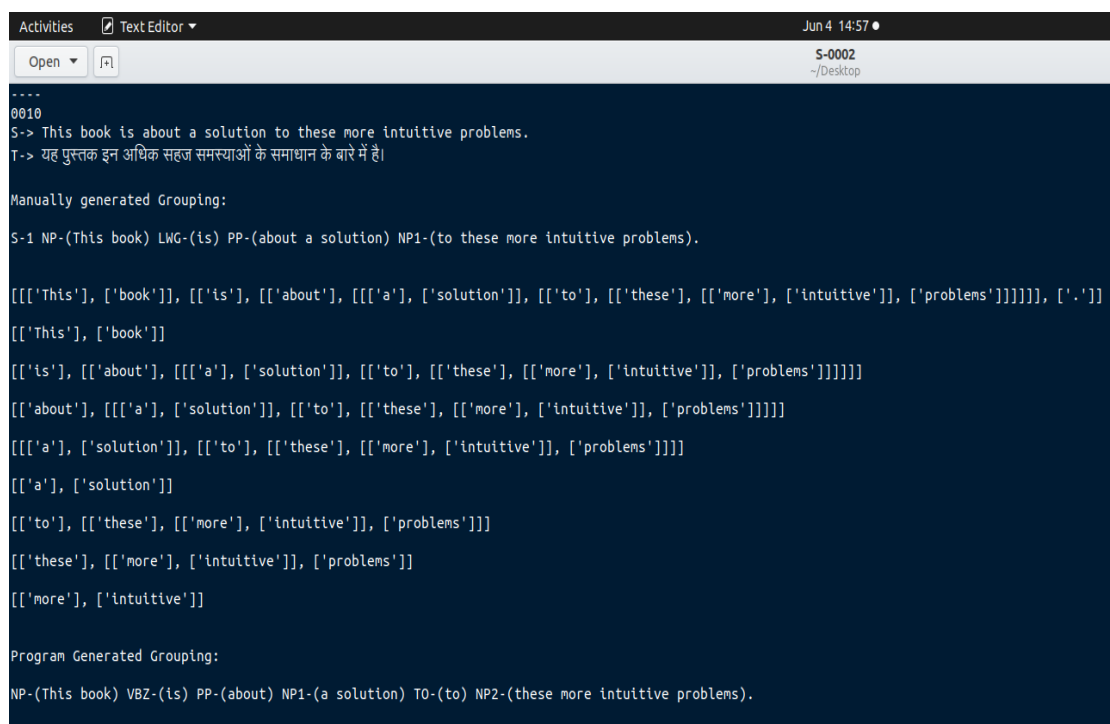
Here the output of sent2table.py in which the Constituency parse of the sentences is converted into a table based on each word's id, POS tag, level, and the word itself. We have compared this with the original Constituency parser tree from the Stanford Constituency parser.

5.3 UNIT TESTING

UNIT TESTING is a level of software testing where individual units/ components of the software are tested. ... A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time



```
Activities Text Editor Jun 4 14:57
S-0002 ~/Desktop

----
0010
S-> This book is about a solution to these more intuitive problems.
T-> यह पुस्तक इन अधिक सहज समस्याओं के समाधान के बारे में है।

Manually generated Grouping:
S-1 NP-(This book) LWG-(is) PP-(about a solution) NP1-(to these more intuitive problems).

[[['This', 'book']], [['is'], ['about'], [['a', 'solution']], [['to'], ['these'], ['more'], ['intuitive']], ['problems']]]], ['.']]
[['This', 'book']]
[['is'], ['about'], [['a', 'solution']], [['to'], ['these'], ['more'], ['intuitive']], ['problems']]]]
[['about'], [['a', 'solution']], [['to'], ['these'], ['more'], ['intuitive']], ['problems']]]]
[['a', 'solution']], [['to'], ['these'], ['more'], ['intuitive']], ['problems']]]]
[['a', 'solution']]
[['to'], ['these'], ['more'], ['intuitive']], ['problems']]
[['these'], ['more'], ['intuitive']], ['problems']]
[['more'], ['intuitive']]

Program Generated Grouping:
NP-(This book) VBZ-(is) PP-(about) NP1-(a solution) TO-(to) NP2-(these more intuitive problems).
```

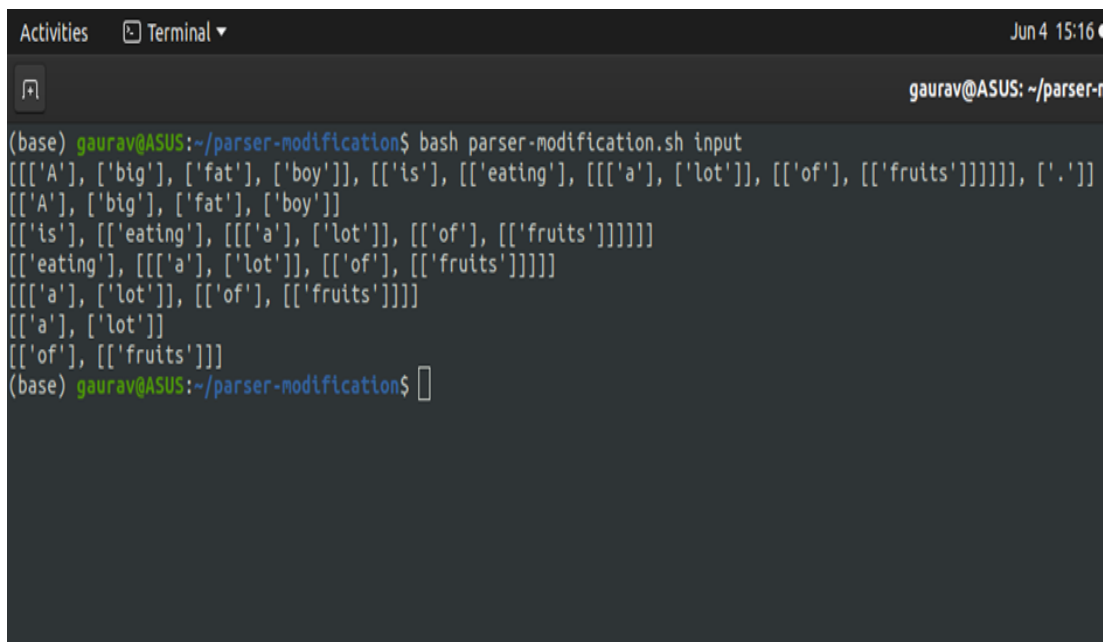
Fig: 5.3 Unit testing of manually and automatically generated grouping

Here the output from parse-modification.py is compared with the manual grouping of a sentence.

5.4 INTEGRATION TESTING

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

As This is output from parse-modification.py as it is based on Constituency parse obtained from Stanford Constituency Parser we have compared it with manual grouping and we found out that in some cases Sanford Constituency Parser fails to generate correct parse so now we are comparing parse from Sanford Parser with manually annotated Lucy treebank to understand in which case Stanford Constituency Parser fails and how we can handle those cases.

A terminal window titled 'Terminal' with a dark background. The prompt is '(base) gaurav@ASUS: ~/parser-modification\$'. The user has entered 'bash parser-modification.sh input'. The output is a list of nested lists representing a constituency parse tree for the sentence 'A big fat boy is eating a lot of fruits'. The output is as follows:

```
(base) gaurav@ASUS:~/parser-modification$ bash parser-modification.sh input
[[['A', ['big', ['fat', ['boy']], ['is', [['eating', [['a', ['lot']], ['of', [['fruits']]]]]], ['.']]
[['A', ['big', ['fat', ['boy']]]
[['is', [['eating', [['a', ['lot']], ['of', [['fruits']]]]]]]
[['eating', [['a', ['lot']], ['of', [['fruits']]]]]
[['a', ['lot']], ['of', [['fruits']]]]
[['a', ['lot']]]
[['of', [['fruits']]]]
(base) gaurav@ASUS:~/parser-modification$
```

Fig: 5.4 Integration testing of the output of Parser-modification.sh

6. CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The work done till the grouping of sentences revealed that the parse generated by the Stanford parser might be incorrect, which directly affects the process of grouping.

The main goal of the project is to develop a tool/software which performs automatic word alignment of a parallel corpus of different language. The work done by us during this project will be a stepping stone for others to carry this research work to achieve the ultimate goal.

6.2 FUTURE SCOPE

The project is a part of research at Anusaarka Lab of International Institute of Information Technology, Hyderabad for their indigenously developed open-source language translator tool.

Our work is the first step in the process to incorporate these methods and many more like these to create a system for alignment of words/phrases in sentences and then these systems would be integrated to build a robust open-source Language Translator tool mainly focusing on the translation of Indian languages to foreign languages and vice-versa.

7. REFERENCES

- [1] Srivastava, Jyoti, and Sudip Sanyal. "POS-based word alignment for small corpus." In 2015 International Conference on Asian Language Processing (IALP), pp. 37-40. IEEE, 2015.
- [2] A Survey Paper on Performance Improvement of Word Alignment in English to Hindi Translation System by Kamala Kant Yadav and Dr. Umesh Chandra Jaiswal.
- [3] Word alignment using Bootstrapping method in Phrase-Based Machine translation by Santanu Pal, Aniruddha Gosh, and Sivaji Bandyopadhyay.
- [4] A Relationship: Word Alignment, Phrase Table, and Translation Quality.
- [5] A survey on parallel corpora alignment (2011)" by Santos, André.
- [6] Word alignment using Bootstrapping method in Phrase-Based Machine translation by Santanu Pal, Aniruddha Gosh, and Sivaji Bandyopadhyay
- [7] By Xin Lian, Kshitij Jain, Jakub Truszkowski, Pascal Poupart, and Yaoliang Yu (2020) "Unsupervised Multilingual Alignment using Wasserstein Barycenter"
- [8] [Alaux et al., 2019] Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. Unsupervised Hyper-alignment for Multilingual Word Embeddings. In ICLR, 2019.
- [9] [Alvarez-Melis and Jaakkola, 2018] David Alvarez-Melis and Tommi S Jaakkola. Gromov-Wasserstein Alignment of Word Embedding Spaces. In ACL, 2018.
- [10] [Artetxe et al., 2017] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning bilingual word embeddings with (almost) no bilingual data. In ACL, 2017.

- [11] [Artetxe et al., 2018a] Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Generalizing and Improving Bilingual Word Embedding Mappings with a Multi-Step Framework of Linear Transformations. In AAAI, 2018.
- [12] Word Alignment by Re-using Parallel Phrases by Maria Holmqvist, December 2008, ISBN 978-91-7393-728-3, Linköping Studies in Science and Technology.
- [13] Shweta Dubey and Tarun Dhar Diwan “Supporting Large English-Hindi Parallel Corpus using Word Alignment”. International Journal of Computer Application (0975-8887) Volume 49-no. 6, July 2012.
- [14] A Survey Paper on Performance Improvement of Word Alignment in English to Hindi Translation System by Kamala Kant Yadav and Dr. Umesh Chandra Jaiswal.
- [15] Chatterjee N and Agrawal S. (2006) Word Alignment in English Hindi Parallel Corpus Using Recency-Vector Approach: Some Studies. 21st International Conference on Computational Linguistics, Sydney, Australia, pp. 51-60.
- [16] Aswani N and Gaizauskas R. (2005a) A hybrid approach to align sentences and words in English-Hindi parallel corpora. In Proceedings of the ACL Workshop on Building and using parallel texts, Ann Arbor, pp. 57-64.

ANNEXURE I

1. Stakeholder Detail

Project Title: Word Phrase Alignment

Project Type: Industry

Stakeholders:

Sr. No.	Stakeholder	Name	Designation, Company name	Contact No.	E-mail Id
1	Industry Mentor	Dr. Vineet Chaitanya	Distinguished Faculty, IIIT Hyderabad	4066531335	Vc9999999@gmail.com
2	Industry Mentor	Dr. Soma Paul	Research Assistant Professor, IIIT Hyderabad	9912910148	soma@iiit.ac.in
3.	Project Guide	Prof. Pooja Walke	Assistant Professor, SVPCET Nagpur	9766249101	pwalke@stvinc.entngp.edu.in
4.	Alumni Mentor	CBS Manaswini	Associate System Engineer Trainee, TCS	7757802425	cmanaswini29@gmail.com

Project Members Details:

Sr. No.	Name	Contact No.	Email Id	Role	Github Account Details
1.	Gaurav Bhisikar	7798827901	gaurav.bhisikar@gmail.com	Group Leader, Developer	https://github.com/gauravbhisikar
2.	Aditya Dale	9503553426	adityadale39@gmail.com	Project Analyst	https://github.com/AdityaDale
3.	Sagar Malik	9370756840	shorya848army@gmail.com	Developer	https://github.com/Sagar8Malik

Project Specific PO Mapping

Program Outcomes	Weak	Moderate	Strong
PO1 (Engineering Knowledge)		✓	
PO2 (Problem Analysis)			✓
PO3 (Design/development of solutions)			✓
PO4 (Conduct investigations of complex problems)			✓
PO5 (Modern tool usage)			✓
PO6 (The engineer and society)		✓	
PO7 (Environment and sustainability)		✓	
PO8 (Ethics)		✓	
PO9 (Individual and team work)			✓
PO10 (Communication)			✓
PO11 (Project management and finance)		✓	
PO12 (Life-long learning)			✓

Subject Mapping:

Artificial Intelligence and Software Engineering

Quality of Project

☒ Environment

☐ Safety

☐ Ethics

☐ Cost

Type of Project

☐ Application

☐ Product

☒ Research

☐ Review