St. Vincent Pallotti College of Engineering and Technology

DEPARTMENT OF INFORMATION TECHNOLOGY

Academic Year 2020-21

Final Project Seminar

on

“Word Phrase Alignment”

Name of Industry: IIIT Hyderabad
Industry Mentor: Dr. Soma Paul(Professor)
Name of Guide: Prof. Pooja Walke(Asst. Prof.)
Alumni Mentor: CBS Manaswini

Project Member:
Gaurav Bhisikar
Aditya Dale
Sagar Malik

Table of Content

Introduction

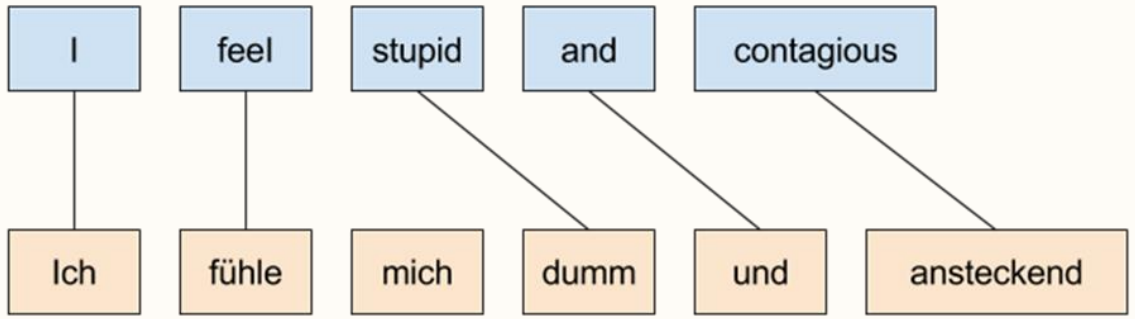
- We are working on Word Phrase Alignment of parallel text.
- **Word Phrase Alignment** is the process of establishing the better translation relationship between the words of a parallel or bilingual corpus.
- Our Project focuses on extracting translation information from parallel text and to determine which words in the source text correspond to which words in the target text.

What is Word Phrase Alignment?

- Word-Phrase alignment is mapping of words between two sentences that have the same meaning in two different languages.
- In simple words, It is sequencing or arranging of words of the translated sentence in such a manner the meaning of the original and translated sentence remains the same.

What is Word Phrase Alignment?

- In the German translation of the line from “Smells Like Teen Spirit” by Nirvana, the word mich doesn’t have a correspondent word in the original version.



Overview

- **Phrase:** A Phrase is a group of words that is part of, rather than the whole of, a sentence.
- **Parallel Text:** A parallel text is the set formed by a text and its translation (in which case it is called a bi-text) or translations.
- **Constituency Parse:** Constituency parsing is the task of breaking a text into sub-phrases, or constituents.

Literature Survey

Sr. No	Title of paper	Author Name	Year	Findings
1	**Unsupervised Multilingual Alignment using Wasserstein Barycenter**	**Xin Lian, Kshitij Jain, Jakub Truszkowski, Pascal Poupart, and Yaoliang Yu**	**2020**	**This paper focuses on unsupervised multilingual alignment, the problem of finding word-to-word translations between multiple languages without using any parallel data.**
2	**Unsupervised Hyper-alignment for Multilingual Word Embeddings. In ICLR, 2019.**	**Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin.**	**2019**	**This paper extends this line of work to the problem of aligning multiple languages to a common space. A solution is to independently map all languages to a pivot language. Unfortunately, this degrades the quality of indirect word translation. We thus propose a novel formulation that ensures composable mappings, leading to better alignments.**

Literature Survey

Sr. No	Title of paper	Author Name	Year	Findings
3	Alignment of Word Embedding Spaces. In ACL, 2018.	David Alvarez-Melis and Tommi S Jaakkola. Gromov-Wasserstein	2018	In this paper, the author cast the correspondence problem directly as an optimal transport (OT) problem, building on the idea that word embeddings arise from metric recovery algorithms.
4	"POS-based word alignment for small corpus." In 2015 International Conference on Asian Language Processing (IALP), pp. 37-40. IEEE, 2015	Srivastava, Jyoti, and Sudip Sanyal.	2015	This paper describes the use of POS tags to enhance the performance of statistical word alignment. The approach shown in this paper works with the small size of the corpus. The authors conducted Experiments on a corpus of 1000 sentences

Problem Definition

- Current deep learning models are trained by parallel text corpus that are on sentence level, but it requires a very large size corpus to train and it takes a lot of time.
- So what we are trying to do is bring the parallel text corpus on Word Phrase Level for training deep learning models, it will require a small corpus to train and it will be faster and more accurate.

Proposed System

- The proposed system is to build an automatic Word Phrase Alignment Tool with high accuracy and efficiency.
- The proposed system will generates parallel corpus break down to word-phrase level.
- Then this parallel corpus will be used to train deep learning models.

Hardware & Software Requirements

Hardware Requirements:

- Intel Core i3 or above
- 4 GB RAM or more
- 100 GB Storage or more
- GPU- Integrated or Dedicated(optional)

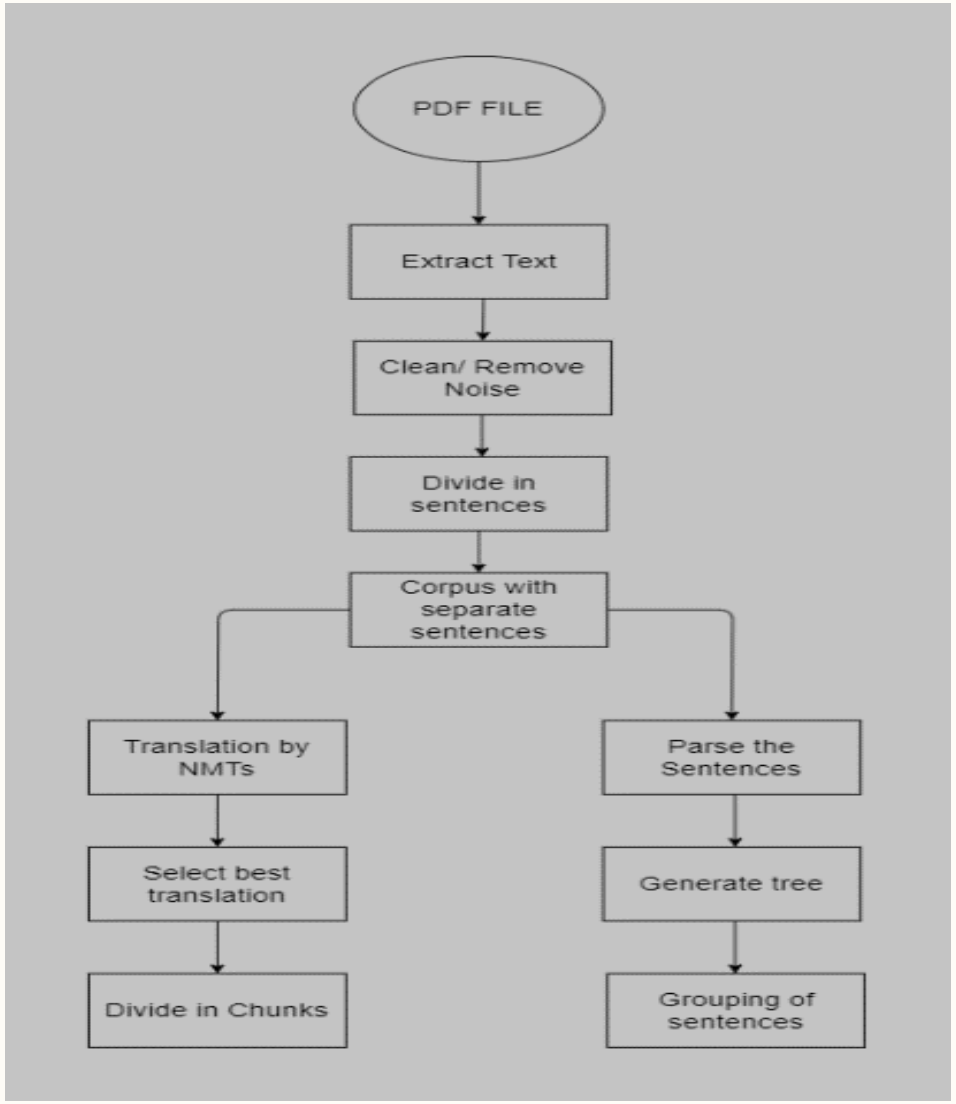
Software Requirements:

- Ubuntu OS
- Python3
- UCCA
- ACE Parser
- Stanford Parser

Technology Platform Overview

- We are working on the technology of NLP.
- NLP is a form of AI that extracts meaning from human language to make decisions based on the information.
- NLP is behind the scenes for several things you may take for granted every day. When you ask Siri for directions or to send a text, NLP enables that functionality.

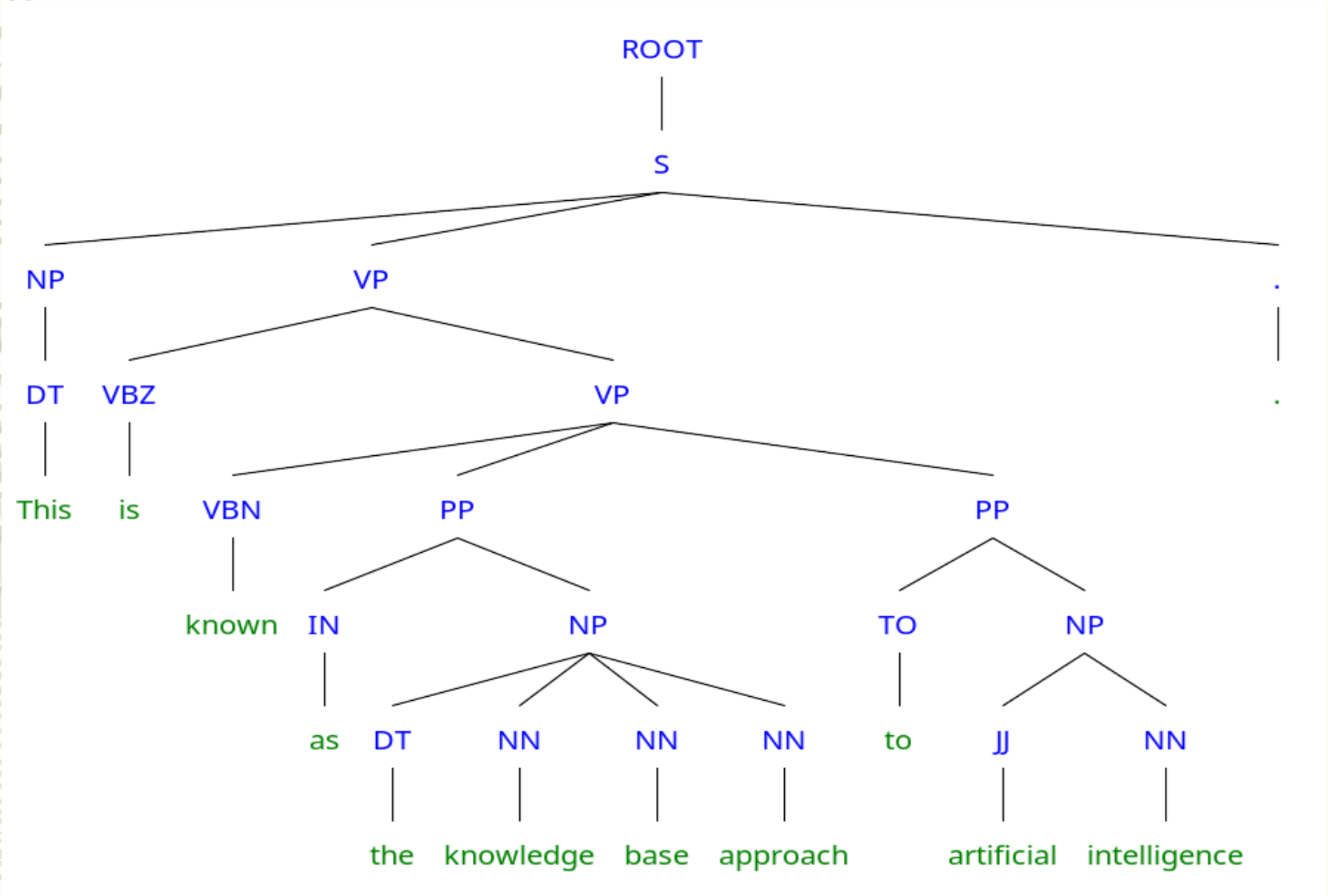
Project Flow

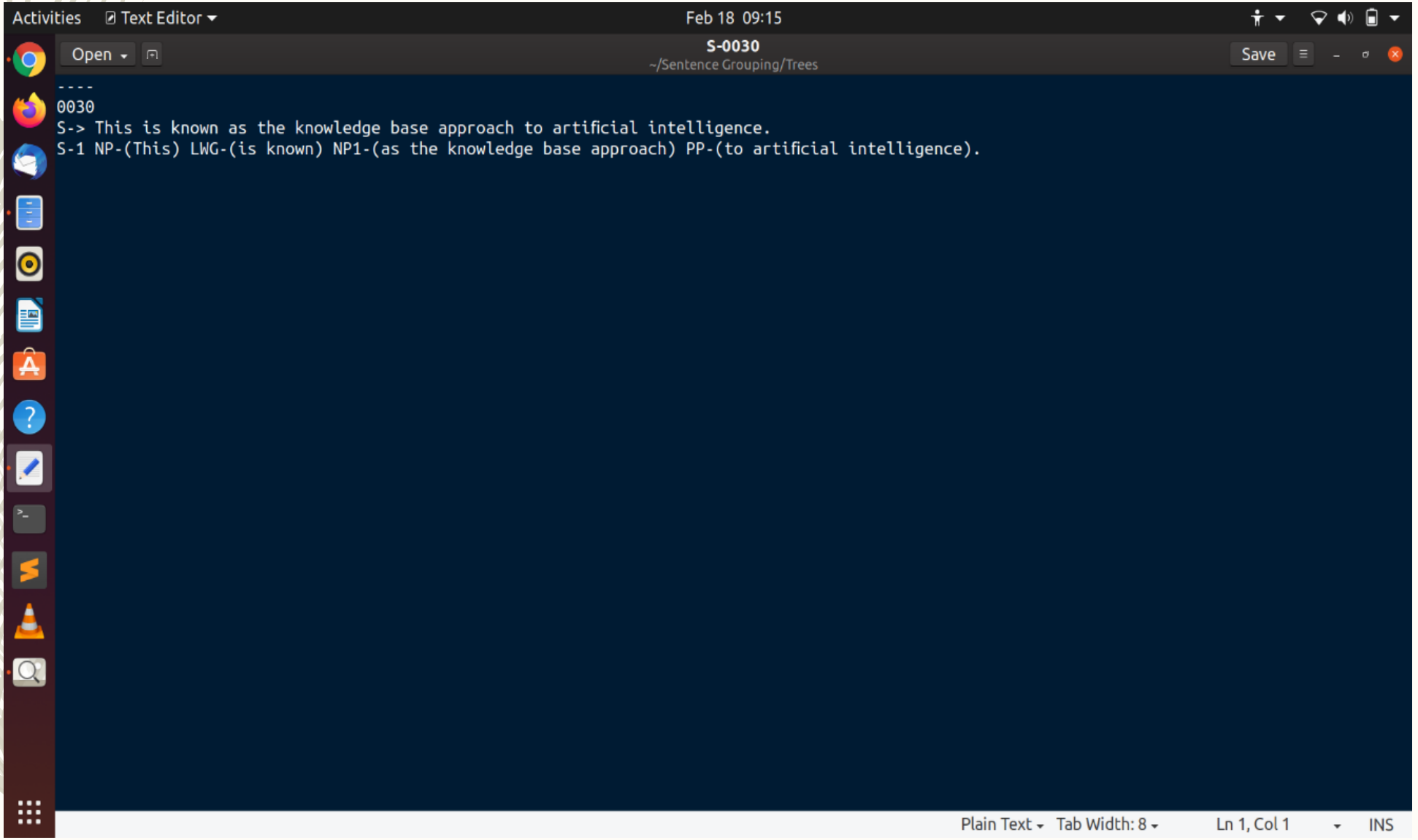


Grouping of Sentences

- Grouping of sentence means labeling parts of sentence in NP, VP, VBZ, PP, etc.
- For Grouping Sentence is first parsed using Stanford parser.
- Then using the constituency parse we generate the tree, with reference to the tree we group the sentence.

Example of Grouping




Activities
Text Editor
Feb 18 09:15
Open
S-0030
~/Sentence Grouping/Trees
Save

0030
S-> This is known as the knowledge base approach to artificial intelligence.
S-1 NP-(This) LWG-(is known) NP1-(as the knowledge base approach) PP-(to artificial intelligence).
Plain Text
Tab Width: 8
Ln 1, Col 1
INS

Program Noise Remove

- When we extract text directly from PDF's of books, etc. it also contains unwanted sentences, words and other information.
- We need clean corpus for translation without any beside the sentences we want to translate.
- So we are using this python script for extraction and removal of noise from text.

Snapshot of Program

```
# USAGE
# python removeNoise.py fileName


import sys

file = sys.argv[1]


def combChapterInt(sentence):
        intgers = ['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 intgers):
                        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 intgers):
                        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()
```

Program Noise Remove

- This Program is used to remove noise or unwanted text/data from a corpus/text file.
- Example: **“INTRODUCTION”**, **“CHAPTER NO.”**, **“PAGE NO.”** and more unwanted text.
- And the output we will get will be the text of the book without any noise or irrelevant data.

Input

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 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
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

Output

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
25

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

Sent2table.py

- This program helps us to create a table based on level, ID and POS Tag which will be used to automate the process of grouping of sentence.
- This program uses constituency Parse generated by Stanford Parser to get information about the Sentence Tree, such as Level of nodes, POS tag of words and etc.

Sent2table.py

sent2table.py

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

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

for i in parse:
    lexParse += i

# print(lexParse)
# print("-----------------------------------------\n\n")


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


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

Line 12, Column 54

sent2table.py

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

    if(lexParse[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)

# l = {}
```

Line 12, Column 54

Sent2table.py

- This program uses the Constituency parse and converts the data available in it to create a table.
- This table consists data about each nodes of Parse Tree.
- The table consists the following information about each node i.e. ID, level, POS Tag and Word.

Input

Open

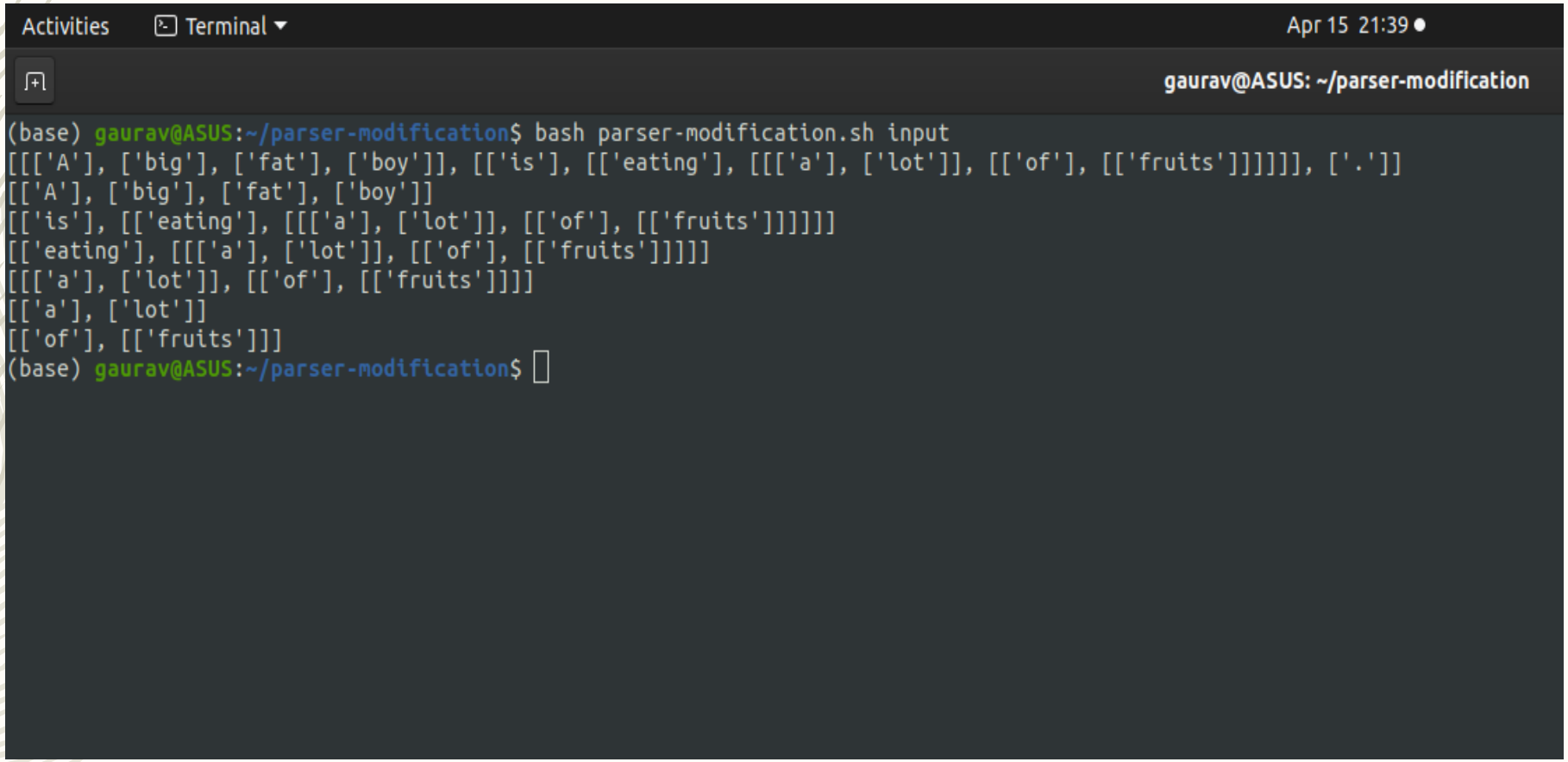
```
(ROOT
  (S
    (NP (PRP We))
    (VP (VBP look)
      (PP (IN to)
        (NP (JJ intelligent) (NN software)))
      (S
        (VP (TO to)
          (VP
            (VP (VB automate)
              (NP (JJ routine) (NN labor)))
            (, ,)
            (VP (VB understand)
              (NP (NN 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)))))))
    (. .)))
```

Output

Activities Terminal

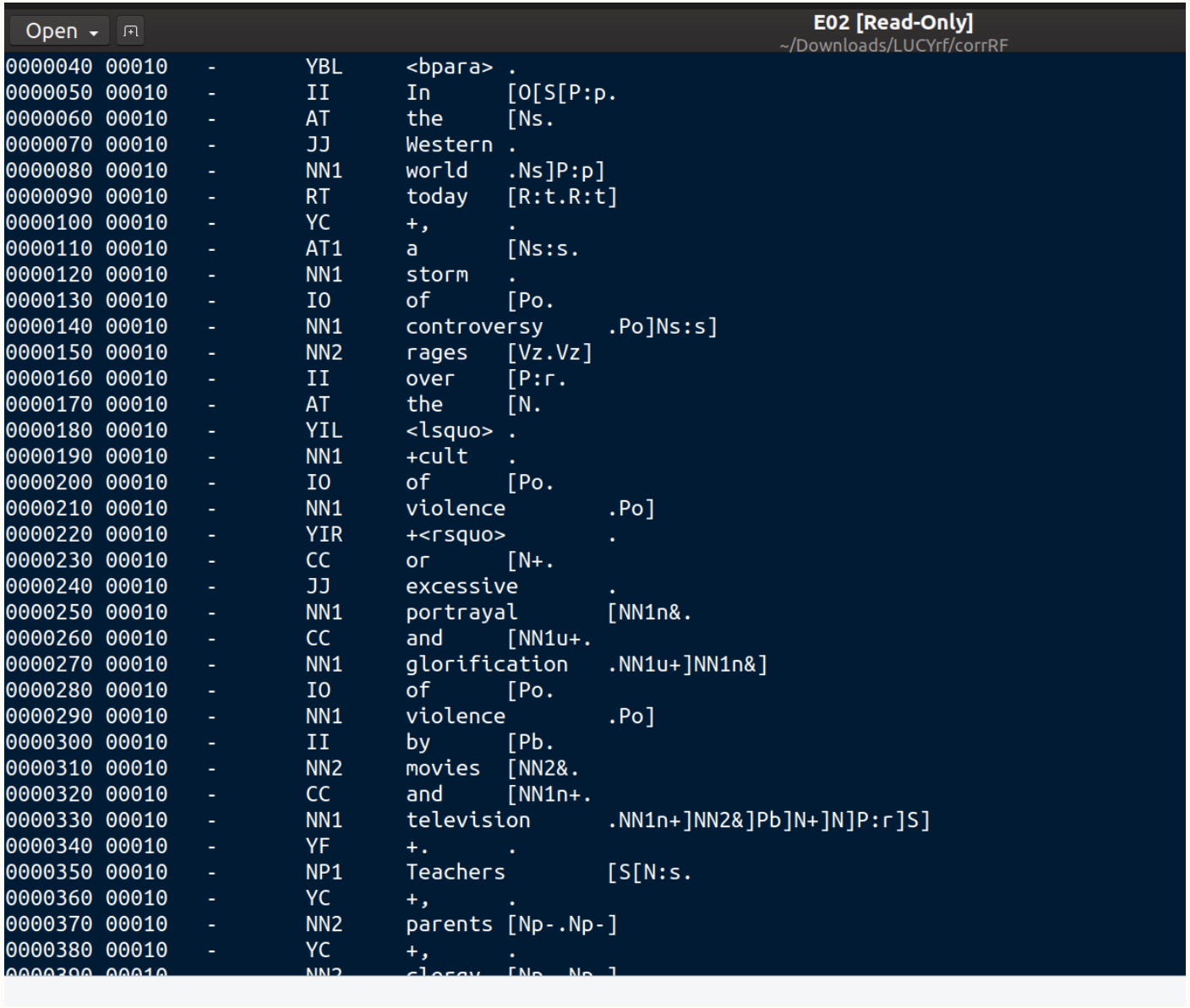
```
(base) gaurav@ASUS:~/Grouping$ python sent2table.py
[0, 'ROOT', 0, '-']
[1, 'S', 1, '-']
[2, 'NP', 2, '-']
[3, 'PRP', 3, ' We']
[4, 'VP', 2, '-']
[5, 'VBP', 3, ' look']
[6, 'PP', 3, '-']
[7, 'IN', 4, ' to']
[8, 'NP', 4, '-']
[9, 'JJ', 5, ' intelligent']
[10, 'NN', 5, ' software']
[11, 'S', 3, '-']
[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$
```

Final Grouping Output



LUCY Treebank

- LUCY Treebank is a collection of manually parsed sentences.
- It is used as a training dataset for parser to learn parsing.
- We used LUCY Treebank to manually generate parse of a sentence and cross check it with parse of Stanford Parser.



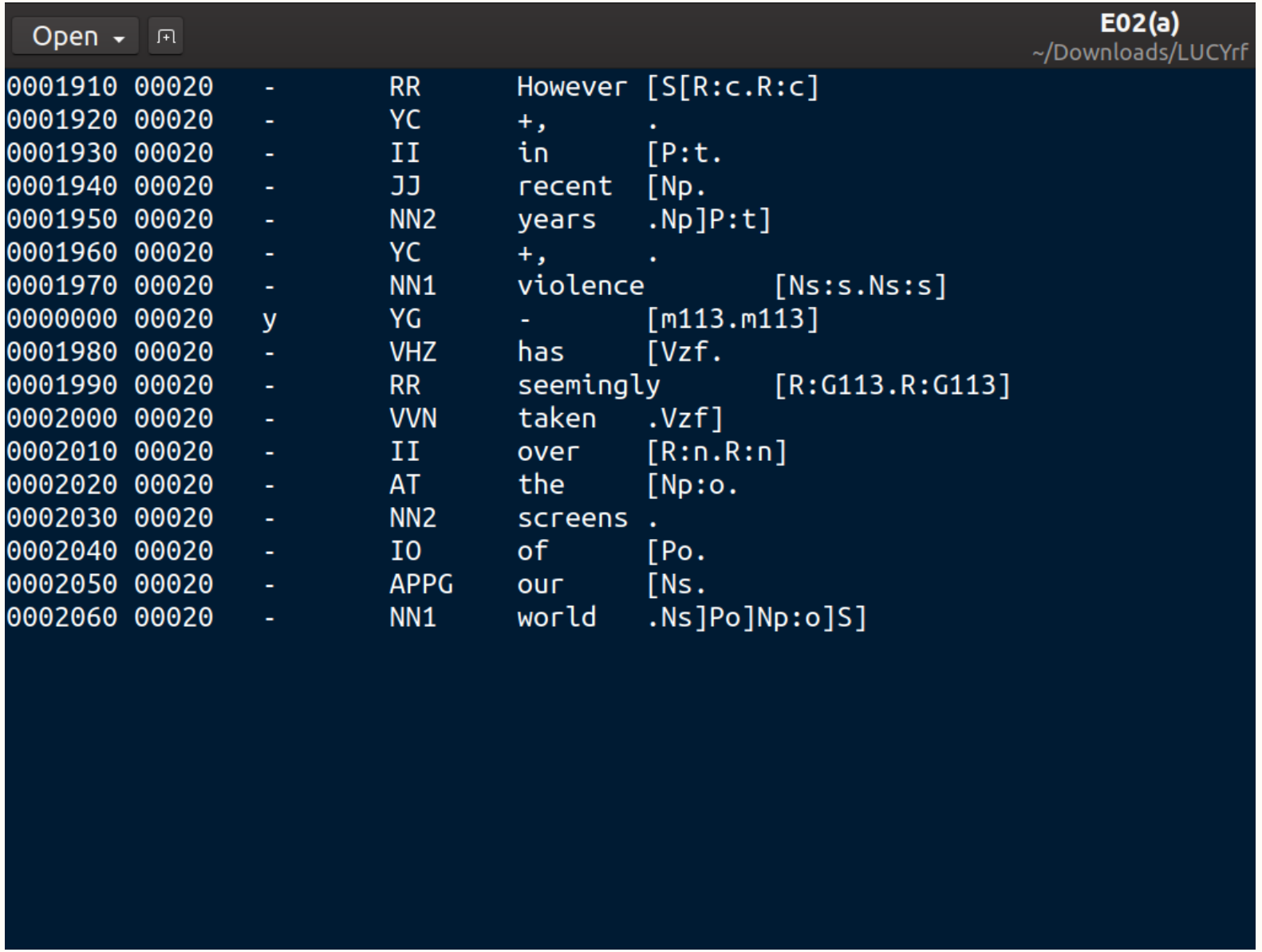
LUCY Treebank

This is the file structure of Lucy Treebank.

6/19/2021

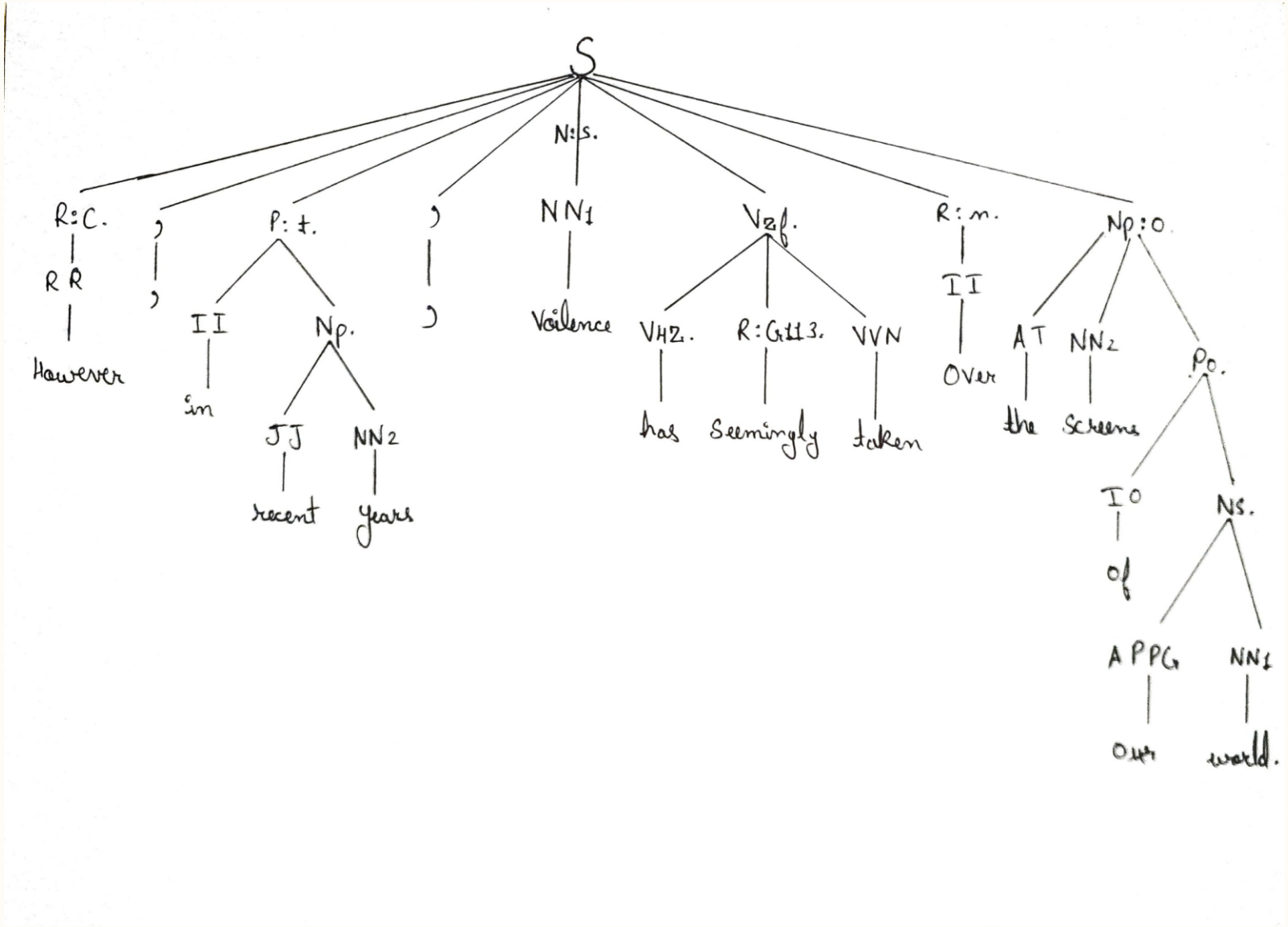
Parsing using LUCY Treebank

- We used sentences in LUCY Treebank, and created their parse tree by hand.
- Then we parsed the same sentence using Stanford Parser and generated it's parse tree.
- Then we compared both the parse trees to find where the Stanford is doing mistake.



Example

For example, we created parse tree for this sentence using the notations in the parse field.



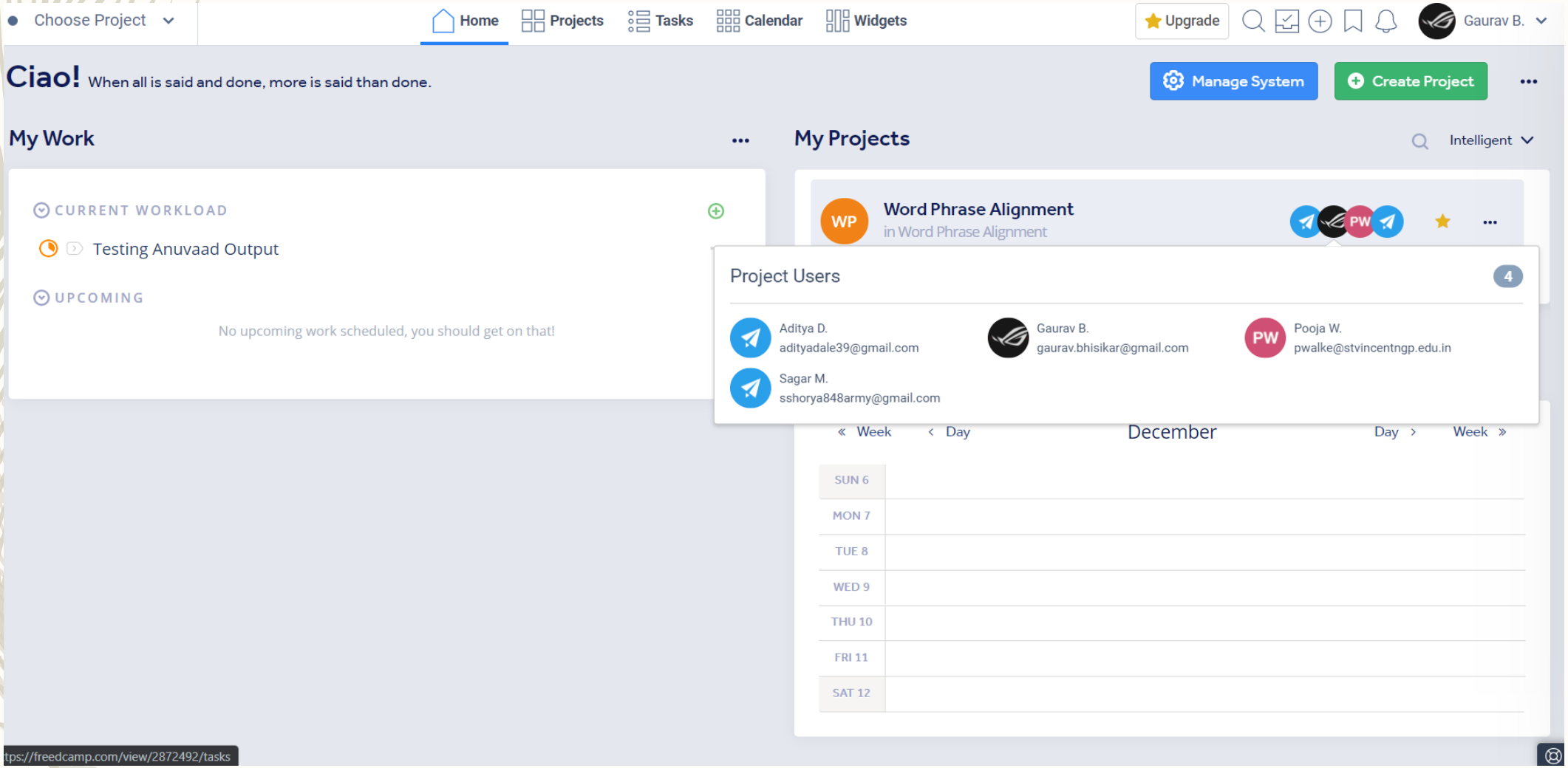
Example

This the handmade parse tree of the sentence shown in the previous slide.

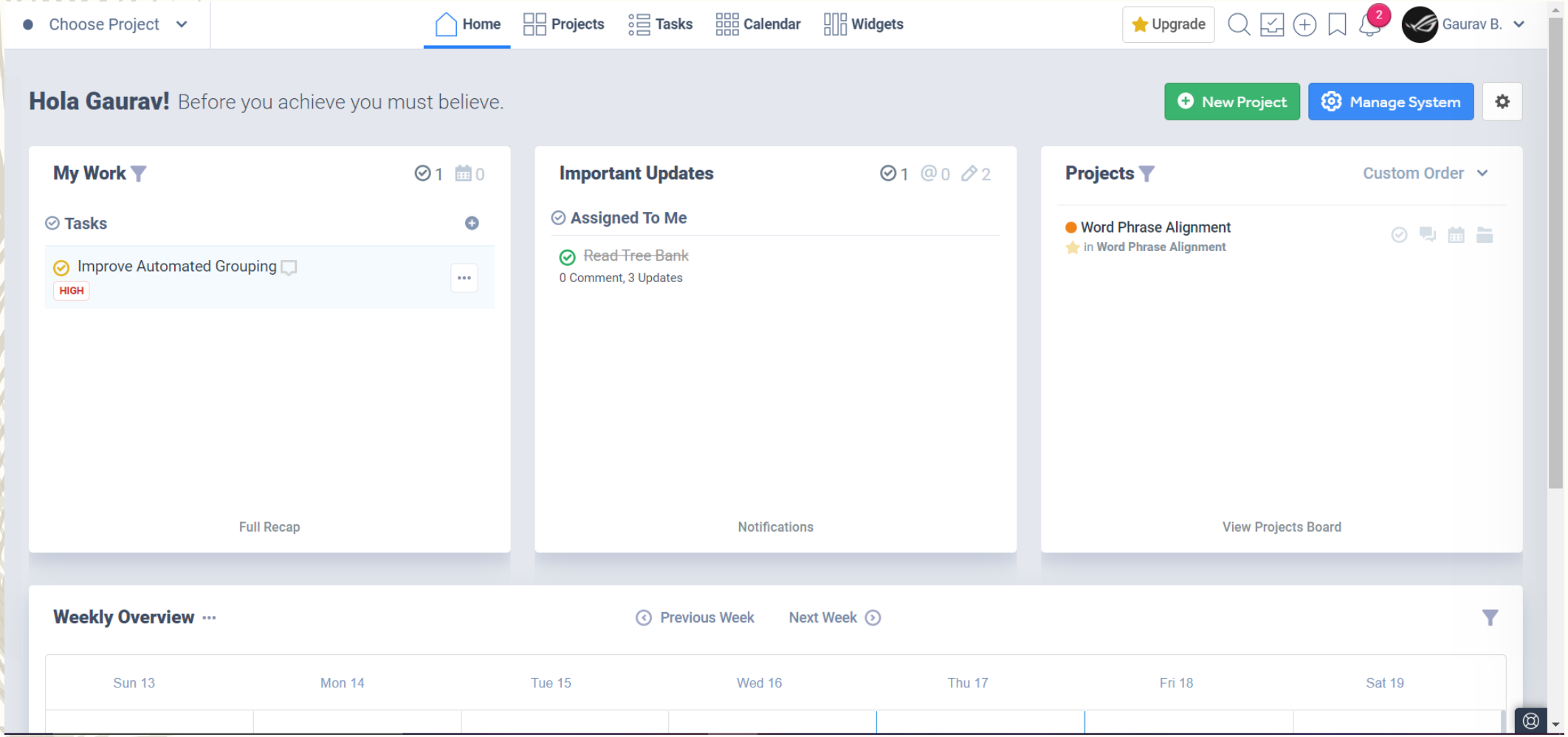
References

- [1] By Xin Lian, Kshitij Jain, Jakub Truszkowski, Pascal Poupart, and Yaoliang Yu (2020) “Unsupervised Multilingual Alignment using Wasserstein Barycenter”
- [2] [Alaux et al., 2019] Jean Alaux, Edouard Grave, Marco Cuturi, and Armand Joulin. Unsupervised Hyper-alignment for Multilingual Word Embeddings. In ICLR, 2019.
- [3] [Alvarez-Melis and Jaakkola, 2018] David Alvarez-Melis and Tommi S Jaakkola. Gromov-Wasserstein Alignment of Word Embedding Spaces. In ACL, 2018.
- [4] 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.

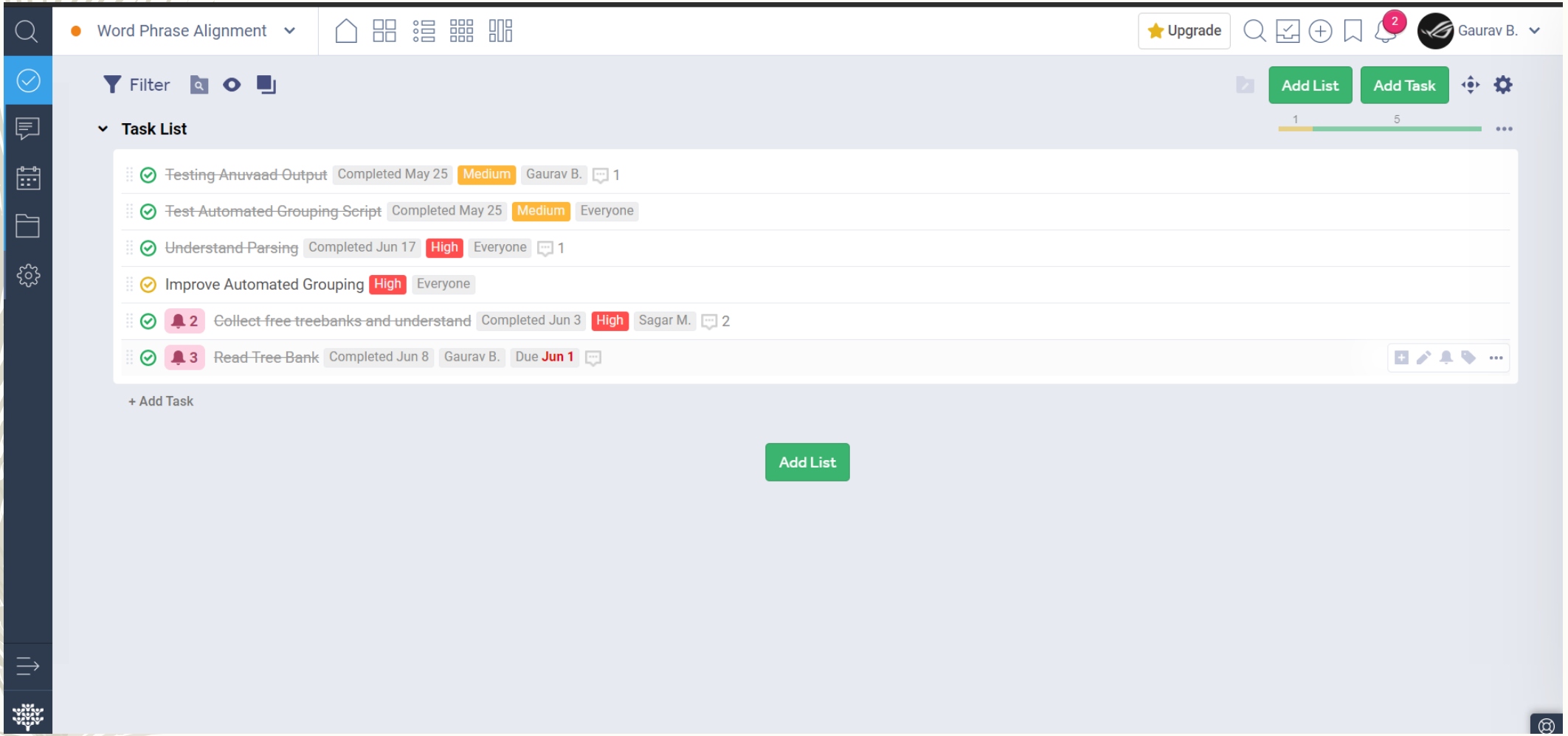
Freed Camp Status



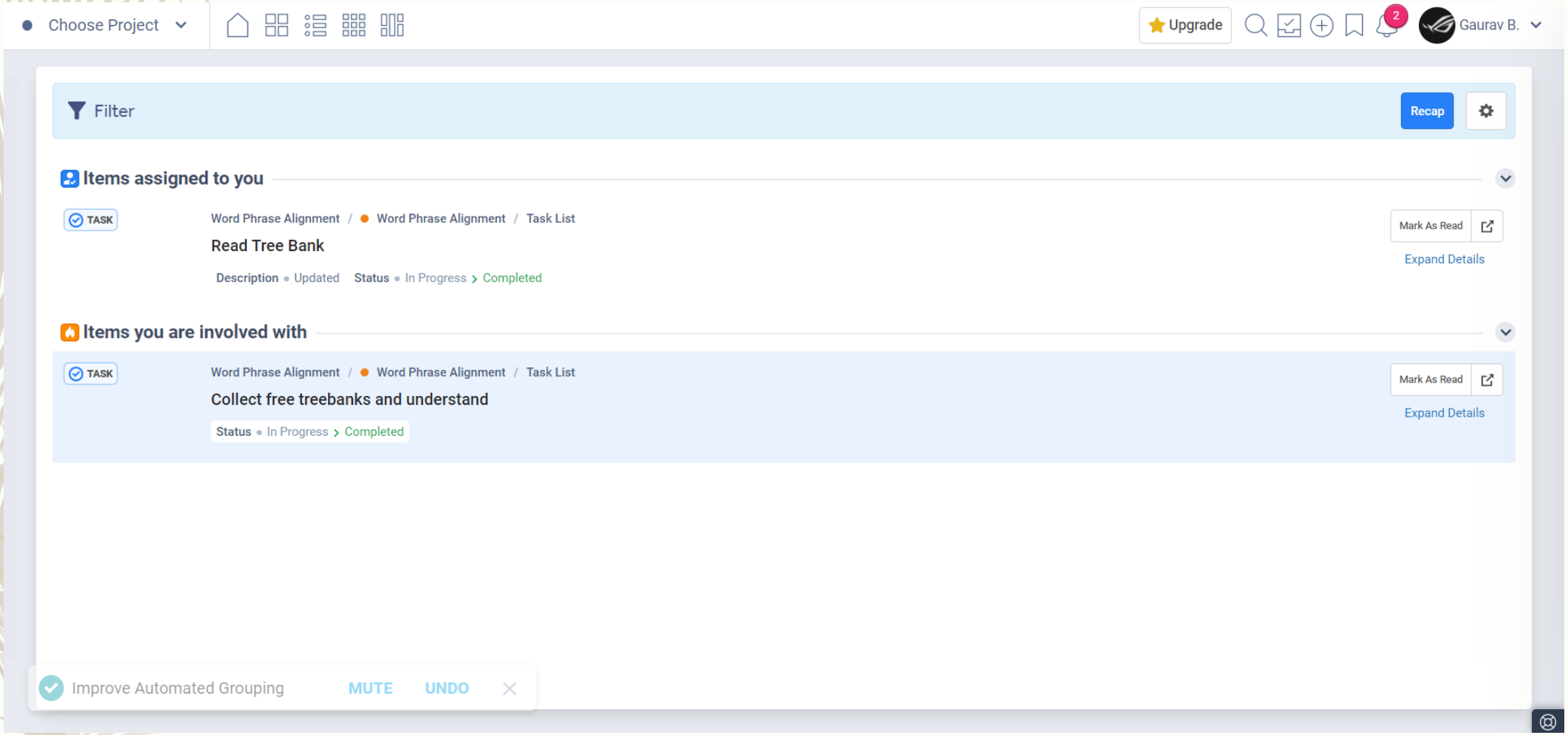
Freed Camp Status



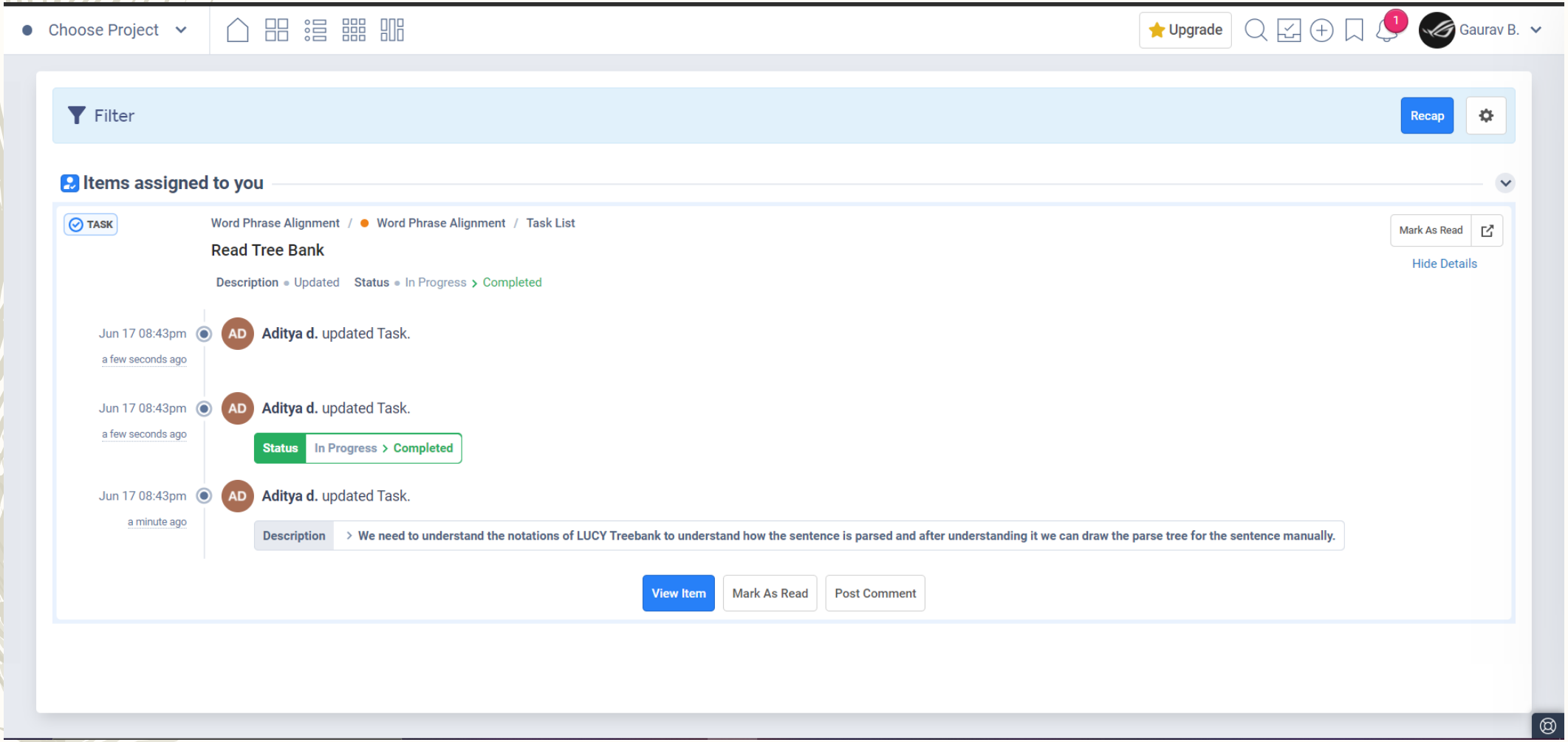
Freed Camp Status



Freed Camp Status



Freed Camp Status



GitHub Account Details

Sr. No.	Name of the students	Account	Link
1	Gaurav Bhisikar	gauravbhisikar	https://github.com/gauravbhisikar
2	Aditya Dale	AdityaDale	https://github.com/AdityaDale
3.	Sagar Malik	Sagar8Malik	https://github.com/Sagar8Malik

THANK YOU!