Semantic Distinguishing to Identify the Context of Words in Telugu Sentences

Major Project Report

Submitted in partial fulfilment of the requirements for the award of the Degree of

Bachelor of Technology (B. Tech)

In

COMPUTER SCIENCE AND ENGINEERING

By

K NEHA	**18AG1A0521**
P SHIVANI	**18AG1A0545**
M INDU	**18AG1A0534**
S ARJUN	**18AG1A0551**
N NIRMALA	**19AG5A0505**

Under the Esteemed Guidance of

Dr. M. V. VIJAYA SARADHI

Professor

Department of Computer Science and Engineering

ACE ENGINEERING COLLEGE

An AUTONOMOUS Institution

NBA Accredited B. Tech Courses, Accorded NAAC 'A' Grade

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana)

Ankushapur (V), Ghatkesar (M), Medchal – Malkajgiri Dist - 501 301.

JULY 2022

ACE Engineering College

An Autonomous Institution

(NBA ACCREDITED B. TECH COURSES: EEE, ECE, MECH, CIVIL & CSE, ACCORDED NAAC 'A' GRADE)

Ghatkesar, Hyderabad- 501 301

(Affiliated to Jawaharlal Nehru Technological University Hyderabad) Website: www.aceec.ac.in E-mail: info@aceec.ac.in

CERTIFICATE

This is to certify that the Major project work entitled **"SEMANTIC DISTINGUISHING TO IDENTIFY THE CONTEXT OF WORDS IN TELUGU SENTENCES"** is being submitted by **K. NEHA(18AG1A0521), P. SHIVANI(18AG1A0545),M. INDU(18AG1A0534),S. ARJUN(18AG1A0551), N.NIRMALA (19AG5A0505) in partial** fulfilment for the award of Degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** to the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2020-21 is a record of Bonafide work carried out by him under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma

Internal Guide
Dr. M. V. VIJAYA SARADHI
Professor and Head
Dept. of CSE

Head of the Department
Dr. M. V. VIJAYA SARADHI
Professor and Head
Dept. of CSE

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We would like to express my gratitude to all the people behind the screen who have helped me transform an idea into a real time application.

We would like to express my heart-felt gratitude to my parents without whom we would not have been privileged to achieve and fulfil my dreams.

A special thanks to our Secretary, **Prof. Y. V. GOPALA KRISHNA MURTHY,** for having founded such an esteemed institution. I am also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project.

We profoundly thank **Prof. DR. M. V. VIJAYA SARADHI**, Head of the Department of Computer Science and Engineering, who has been an excellent guide and a great source of inspiration to my work.

We extremely thank **Mrs. SOPPARI KAVITHA** Associate Professor, Project coordinator, who helped us in all the way in fulfilling of all aspects in completion of our Major-Project.

We are very thankful to my internal guide **Prof. DR. M. V. VIJAYA SARADHI,** Head of the Department of Computer Science and Engineering, who has been an excellent and given continuous support for the Completion of my project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, we would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased our task.

K NEHA (18AG1A0521)
P SHIVANI (18AG1A0545)
M INDU (18AG1A0534)
S ARJUN (18AG1A0551)
N NIRMALA (19AG5A050

DECLARATION

This is to certify that the work reported in the present project titled **"Semantic Distinguishing To Identify The Context Of Words In Telugu Sentences "** is a record work done by us in the department of Computer Science and Engineering, ACE Engineering College.

No part of the thesis is copied from books/journals/internet and whenever the portion is taken, the same has been duly referred in the text; the reported are based on the project work done entirely by us not copied from any other source.

K NEHA (18AG1A0521)
P SHIVANI (18AG1A0545)
M INDU (18AG1A0534)
S ARJUN (18AG1A0551)
N NIRMALA (19AG5A0505)

ABSTRACT

Word Sense Disambiguation (WSD) is an important field in the natural language processing domain. It is mainly used in fields such as Machine Translation, Information retrieval and Text mining. These do not have a built-in word sense disambiguator and thus, it has to be performed explicitly. Up until now, this has been largely performed on sentences in the English language to identify the sense or context of a word in that sentence. This project aims to implement it for regional languages as well, specifically, Telugu. Since, Telugu vocabulary is huge, occurrences of ambiguity is common and this project is aimed at resolving them. To achieve this, a synset will be used, which is an interface for the Telugu WordNet. The algorithm used in this project is the Lesk algorithm which is a classical word sense disambiguation algorithm. It is based on the assumption that the correct meaning/sense of a word is dependent upon the words present in its neighborhood. More specifically, sense or context of a word is determined by comparing the meaning of each sense of the word with the meanings of the neighboring words. This project aims to get the best sense of words based on the neighboring words of a particular word using Lesk algorithm.

Keywords: Word Sense Disambiguation, Telugu, Lesk, WordNet, IndoWordNet

INDEX

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

1.1 Introduction

1.1.1 Word Sense Disambiguation (WSD)

Word Sense Disambiguation (WSD) refers to the process of ascertaining which "sense" of a particular word is used in a sentence based on the context of the sentence. Determining which sense of a word is used in a sentence is very important. Word Sense Disambiguation is used in doing the same [1, 2]. There are numerous applications to Word Sense Disambiguation. Some of them are:

Machine Translation: Some words can have different translations based on different senses of the word. Since a machine cannot implicitly determine the correct meaning of a word based on context, Word Sense Disambiguation can be useful in such cases.

Information Retrieval: As the name suggests, Information Retrieval is a software program that deals with storage and effective retrieval of information from document storage space. WSD can be used to remove ambiguities from queries provided by the user to search the document base.

Text Mining: Word Sense Disambiguation can be used in the context of Text Mining to flag relevant words based on their senses.

1.1.2 WSD for Telugu

Word Sense Disambiguation is the task that automatically identifies the exact sense of the ambiguous word based on the context in which the word is used.

Telugu, which is an Indian regional language, spoken by the people of states of Telangana and Andhra Pradesh. Since Telugu vocabulary is huge, occurrences of ambiguity are common, to achieve this knowledge-based algorithm is used which includes finding the overlap between the features of different senses of ambiguous word and the features of the word in its context. That is performed by taking words in the neighbourhood of the target word and the sense which has the maximum overlap is selected as the context appropriate sense. WSD in telugu is achieved after performing necessary pre-processing steps such as translating the words, stop words filtration, stemming, etc.,

1.2 Existing System in WSD

The present in WSD focus mainly on the English sentences, and it cannot work for the regional languages. It would be a great problem to find the ambiguity for the regional languages.

Eg: The existing Lesk algorithm for the English sentences works as follows: Consider the following two sentences:

1) Tonight, the **band** is playing my favourite music.

2) The **band** of frequencies used by radio is very large.

The word ‘band’ in both sentences means different things. Humans can tell this difference but a program implementing Lesk does the following:

Generate the synsets of band with their meanings:

- Instrumentalists not including string players
- an unofficial association of people or group
- a group of musicians playing popular music for dancing
- a strip or stripe of a contrasting colour or material
- a range of frequencies between two limits
- a driving belt in machinery
- bind or tie together
- attach a ring to the foot of, in order to identify
- a restraint put around something to hold it together

- Match each meaning with the remaining words in the sentence. For the firstsentence, the meaning – 'a group of musicians playing popular music for dancing' matches the maximum number of words – ("playing", "music") and thus is taken as the right sense

- For the second sentence, the meaning – 'a range of frequencies between two limits' matches the maximum number of words – ("frequencies") and thus taken as the right sense

Based on the above implementation of Lesk, it is possible to apply it on Indic regional languages as well.

1.2.1 Drawbacks of Existing System

The drawbacks of the current system include the following:

- Largely applicable to English language: Most of the work on Word Sense Disambiguation is being performed in the English language. All

new algorithms are first applied to the English language and only then they are adapted to other languages slowly.

- The stemming required to perform Word Sense Disambiguation is performed by many python packages or similar only for English. There are no open source stemmers for Indian regional languages, for example, as a python module.

- There is no interface to the IndoWordNet provided by the 'Pyiwn' package which can be used to view the different synsets of a word, its hyponyms, gloss, etc.

1.3 Proposed System

The proposed system involves to extend Lesk to Telugu language as well. Lesk has been largely applied to sentences in the English language. Most of the research has been performed for English WSD but not much work has been done to implement the same for Telugu language sentences.

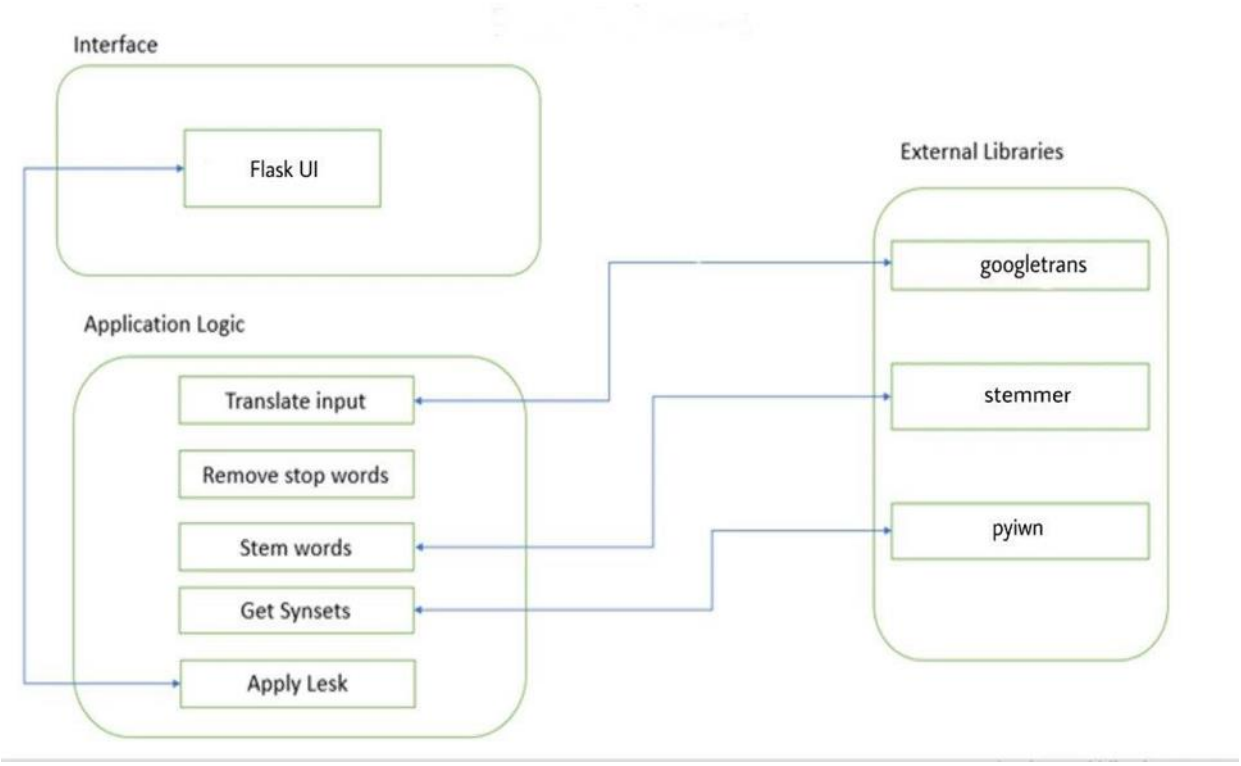


Fig 1.1 WSD for Telugu System Design

The proposed system (*Fig 1*) aims to perform Word Sense Disambiguation for Telugu in the following ways:

- The input is taken in English and translated to Telugu using the python package – googletrans.

Then required pre-processing steps are performed on the sentence :

- Removing stop words
- -Stemming – a stemmer is used to stem words in Telugu
- Next, pyiwn – a python package is used, which provides an API to access the IndoWordNet

- Lastly, the relevant synsets are used and provided as input to the Lesk algorithm

1.3.1 Advantages of Proposed System

The advantages of the Proposed System include the following:

- Applicable for regional languages.
- Word Sense Disambiguation applied for Telugu language by implementing Lesk. All the pre-processing steps required are performed in the Telugu sentence and not on English sentences..
- A stemmer to stem Telugu words is developed to convert most of the common words into their root form. By doing so, we don't have to translate the stemmed English words to Telugu, which leads to loss of semantic meaning of the sentence.
- Providing am interface to the 'Pyiwn' IndoWordNet, using which a user can:

    1. Select any of the available languages.
    2. View all the synsets of a word in the language selected.
    3. View the hyponyms of a selected synset.
    4. View an example of the selected synset used in a sentence.

CHAPTER-2

FEASIBILITY STUDY

A feasibility study is used to determine whether idea works successfully, such as ensuring how a project is legally and technically feasible and economically reasonable, with the rise in the responsibility of project management, there is also a necessity for feasibility study. Usually, feasibility studies is performed before project implementation and technical development.

A well designed feasibility study should have the background of a project, such as the description of a project and it also evaluates the project's potential for success.

For any project to be accepted as feasible, it must satisfy certain parameters such as:

1) Not compromise on the functionality of the project to conform to feasibility parameters defined by the company individual.
2) Adhere to mandatory parameters.

Some of the feasibility parameters are

2.1 Technical Feasibility

Technical feasibility study has the details of delivering a project and also evaluates the hardware and software requirements of the proposed system.[4]

In this project, we require packages like googletrans, pyiwn, nltk, etc. These packages play a major role in fulfilling the project requirements with good accuracy and within minimal time. The languages used are HTML, CSS for Front end development; we also developed an UI using Flask and back-end using python.

2.2 Economic Feasibility

Economic Feasibility is the process of calculating the cost and benefits of a project before a budget is allocated . This is helpful in getting an idea about a particular project, i.e., if it is feasible to go forward with the project in terms of its economic overview.

The project would not require any cost because it is an open source software. The packages used helps in understanding the user without any investment. The main module helps in converting a given English sentence into telugu and also an indo wordnet interface using this we can find out the synsets of the given sentence in any language by selecting the select any language option.

2.3 Legal Feasibility

It ensures legal data access and gives prominence to data security. Since this project does not involve any confidential or protected data about the user, it violates no legal parameters.[4]

2.4 Operational Feasibility

Operational feasibility measures how well a proposed system solves the problem and how it satisfies the requirement identified in the phase of system development. [4]

It minimizes the drawbacks of the current system by aiming at performing word sense disambiguation on Telugu sentences to help gauge the theme of the sentence.

2.5 Scheduling Feasibility

Defined as the probability of a project to be completed within its scheduled time limits, by a planned due date. If a project has a high probability to be completed on-time, then its schedule feasibility is appraised as high. In many cases a project will be unsuccessful if it takes longer than it was estimated: some external environmental conditions may change, hence a project can lose its benefits, expediency and profitability. If a work to be accomplished at a project does not fit the timeframes demanded by its customers, then a schedule is unfeasible (amount of work should be reduced or other schedule compression methods applied).

CHAPTER-3

LITERATURE SURVEY

The purpose of Word Sense Disambiguation is to determine which significant word is triggered by use of the word in particular sentence, different words have different meanings based on the context it used in that sentence. In this case the regional languages may have many words that can be interpreted in different ways depending upon the context of its use. The problem of resolving semantic ambiguity is known as Word Sense Disambiguation.

The major complication of WSD is that to choose the sense of the word because different sense is closely related.

3.1 Research Papers

The research papers and their brief summaries are as follows:

1. J.Sreedhar *et.al,* in their paper spoke about the various methods which can be used to perform Word Sense Disambiguation for Telugu. The paper began by first talking about the taxonomy of Word Sense Disambiguation in NLP and explained in brief about each of them. In their own words, Word Sense Disambiguation is defined as follows:

"Word sense disambiguation (WSD) is the ability to computationally determine which sense of a word is activated by its use in a particular context."

In the following sections of the paper, current state of the art methods used for Word Sense Disambiguation are briefly explained. Some of the methods discussed in the paper are:

- Method proposed by Walker, in which a thesaurus is used. Each word is assigned a particular subject category in the thesaurus. Each subject assigned to a particular word is assumed to be a particular sense and WSD is performed based on this assumption.

- Method proposed by Quillian in mid 1960s in which he proposed the use of semantic network representation of a machine-readable dictionary. In this, a node represents the meaning, and this node is used to connect words.

Many such methods have been specified in this paper along with a brief description of each method. Lesk is one of the proposed methods which uses a WordNet.

- **Merits:** The merit in this paper is that it gives an overview of the work that has been done in the field of Natural Language Processing (NLP) and more specifically in Word Sense Disambiguation, which is a subset of NLP.

  All of these different methods have been taken from various research papers and presented in concise manner such that a person interested in working on Word Sense Disambiguation can get a clear idea of how to proceed.

- **Improvements in this project:** Even though the paper discusses various methods for Word Sense Disambiguation, it doesn't provide any implementation details about it. Lesk has been chosen to perform WSD in this project as it has not been researched extensively for Telugu, unlike other algorithms.

- **Conclusion:** Many methods were analysed in this paper and provided a concise explanation. Since no implementation details were provided, this project aims at implementing Lesk algorithm described in the paper.

2. Ritesh Panjwani *et.al* [8], in their paper spoke about their python-based Application Programming Interface (API) for Indian WordNets. This can be used as a module by getting it from pypi.org, where it is present as an open-source project. In their own words, the aim of their project:

"With our work, we aim to provide an accessible, robust, easy-to-use API for Indian language WordNets."

The paper starts off by talking related work that already exists, such as 'The Java WordNet Library', which is a Java API to access the WordNet. Their motivation was to create an API for IndoWordNet which was python based so that it could be easily integrated with projects.

Next the paper describes the API design and the different features that can be utilised in this module.

- **Merits:** The paper provides a detailed description of how the pyiwn project was built, its motivation, procedure and utilisation procedures. It also gives examples of its usage. It being a python module, it is easier to integrate into projects as it doesn't require additional overhead.

- **Improvements in this project:** The paper provides details about the pyiwn project but not its implementation in Word Sense Disambiguation. This project uses the python module described in the paper and implements it to achieve Word Sense Disambiguation for Telugu. Since it provides an API to access Telugu IndoWordNet along with other languages, this paper was chosen for this project's implementation.

- **Conclusion:** Finally, by implementing the 'pyiwn' module from the paper , which stands for 'Python Indo Word Net', this project aims to achieve Word Sense Disambiguation for Telugu.

3.2 WordNet

WordNet is a large lexical database for English language. The different parts of speech of English language are clustered into sets of synonyms, called synsets. These generated sets are interlinked by means of lexical relations and conceptual semantics. This orientation of WordNet is the reason for its popularity in natural language processing. It represents a thesaurus since it clusters the words into groups based on their meanings. A WordNet can be used to find the hyponyms of a particular word and also its gloss/meaning.

These features are useful in the process of disambiguating a word in a sentence. Other variants of the WordNet are IndoWordNet, EuroWordNet, BabelNet, Mimida Project, etc.

3.2.1 Indo WordNet

IndoWordNet allows access to multiple Indic languages. The available languages include – Hindi, Bengali, Gujarati, Tamil, Telugu, etc., a total of 18 languages. Like the WordNet, it also provides hyponyms and gloss for the purposes of disambiguation. IndoWordNet is a linked structure of WordNet for major Indian languages. It is mainly developed for the WSD of regional languages. IndoWordNet a linked structure of regional languages WordNet, its cross linkage is helpful for WSD. Antonymy it's a lexical relation indicating opposites. Synoymy this is the relationship between words in a synset. These are the lexical features of IndoWordNet.

3.3 PYIWN

Pyiwn is an python based API to access Indian language WordNet. The language for its implementation is selected as python due to its popularity in use cases relating to natural language processing. Pyiwn provides the access to the synsets and their relational connections, this API will be available on the GitHub. This API module can be implemented in the following way.

All the following examples are given assuming 'pyiwn' is imported in the code.

$ indo_wordnet_obj = pyiwn.IndowordNet(<language>)

The synsets can be accessed in the following way:

$ indo_wordnet_obj.all_synsets()

Using the above function, we can use all available synsets for a particular language.

$ indo_wordnet_obj.synsets(word)

To access synsets of a given word

$ indo_wordnet_obj.all_words()

Returns a list of all the words in a language

3.3.1 SYNSET PROPERTIES

Function to access synsets of a given word

$ synset = indo_wordnet_obj.sysnsets(input_word)[0]

Function used to know about the Part of Speech (POS) tag of a synset

$ synset.pos()

Function used to retrieve head word

$ synset.head_word()

Function to retrieve gloss/definition of a word

$ synset.gloss()

Function to retrieve examples of a synset in a given language

$ synset.examples()

3.4 LESK

The Lesk algorithm is a classical word sense disambiguation algorithm introduced by Michael E. Lesk. It is based on the assumption that the sense of a word is dependent upon the words present in its neighborhood. More specifically, sense or context of a word is determined by comparing the meaning of each sense of the word with the meanings of the neighboring words. This comparison is made by matching the words in meanings and the sense whose meaning matches the maximum number of times is taken as the resultant sense of that particular word in a given sentence

Steps in Lesk Algorithm:

- Consider a word for processing.
- Next, take the remaining words and create a list of words which consist of the gloss and examples of those words. Let's call it 'matcher_list'.
- Remove stop words from these lists to reduce irregular matches
- Now for every synset of the word to be processed, create a similar list of words containing the gloss and examples of that synset. Let's call this list 'synset_list'
- Match the above two list to find overlap of words.
- The synset_list which has the maximum overlap with the matcher_list is taken as the best synset and the gloss of the synset is taken as the best sense.

3.5 Natural Language features

The Natural Language API has several methods for performing analysis and annotation on your text. Each level of analysis provides valuable information for language understanding. These methods are listed below:

- **Sentiment analysis** as the name suggests, is the process of analysing a sentence to identify the opinion or emotion present in that sentence. This is performed to gauge the attitude of the person who has written this sentence. i.e., whether the sentiment is neutral, negative or positive.

- **Entity analysis** scans a given text/sentence and looks for known elements such as popular names, landmarks, etc. and retrieves details about such entities.
- **Syntactic analysis** breaks up the given text/sentence into tokens (individual elements of a sentence, usually just words) and retrieves additional information and details about these tokens.

3.6 FLASK

Flask is a web application framework written in python. Flask is one of the most used Python web application framework. Flask package can be installed from the Python Package Index. Flask is based on Werkzeug WSGI toolkit and jinga2 template engine. Flask is often referred as micro framework, its aim at keeping the core of an application simple and yet very extensible.

CHAPTER-4

ALGORITHM DESCRIPTION

4.1 Lesk Algorithm:

Lesk is one of the classical Word Sense Disambiguation algorithms. It was developed by Michael E Lesk. It is based on the assumption that the sense of a word is dependent upon the words present in its neighbourhood. More specifically, sense or context of a word is determined by comparing the meaning of each sense of the word with the meanings of the neighbouring words.

Steps in Lesk Algorithm:

- Consider a word for processing.
- Next, take the remaining words and create a list of words which consist of the gloss and examples of those words. Let's call it 'matcher_list'.
- Remove stop words from these lists to reduce irregular matches.
- Now for every synset of the word to be processed, create a similar list of words containing the gloss and examples of that synset. Let's call this list 'synset_list'
- Match the above two list to find overlap of words.
- The synset_list which has the maximum overlap with the matcher_list is taken as the best synset and the gloss of the synset is taken as the best sense.

The algorithm can be clearly visualized by this pseudo-code in the next page:

```
function LESK(sentence) returns correct sense of words
result := a dictionary of word and sense
correct-sense :=  most frequent sense for word
max-overlap :=  0
words :=  list of words in sentence
for each word in words
    for each sense in sense(word) do
        matcher :=  list of words created from gloss and hyponyms
        current-overlap := COMPUTE_OVERLAP (matcher, word)
        if current-overlap > max-overlap then
            max-overlap :=  current- overlap
            correct-sense :=  sense
    end
    result[word] := correct-sense
end return(result)
```

4.2 Work Flow

Let us now look at the workflow of the disambiguation process. It is represented as a flowchart in *Fig.4.1*. The different steps involved are:

1. Read Input
2. Translate Input
3. Remove stop words
4. Get synsets
5. Apply Lesk

6. Display results

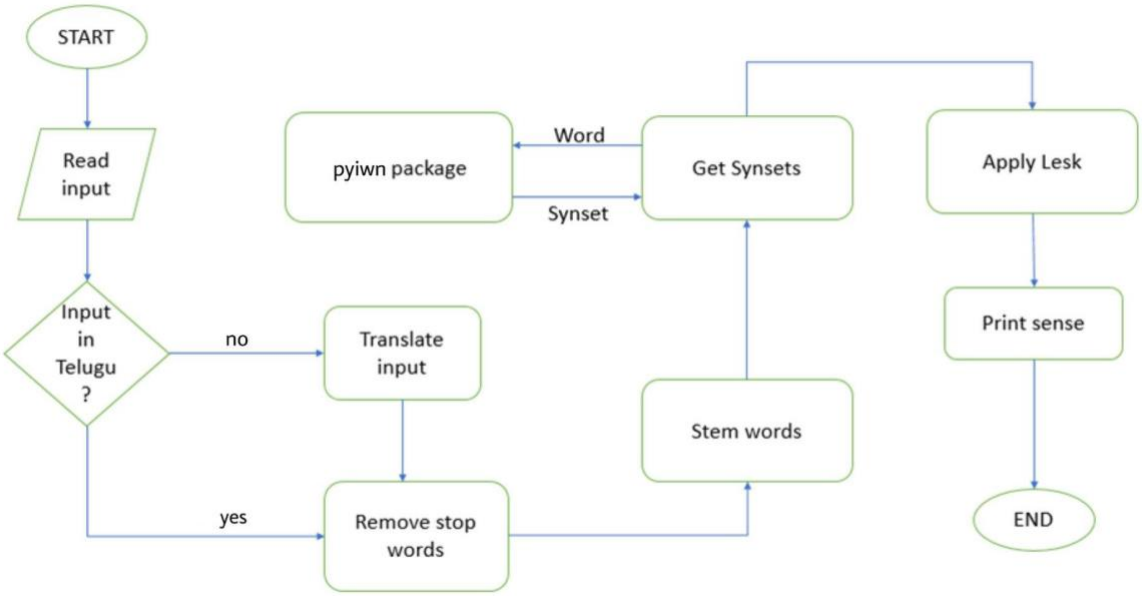


Figure 4.1 : Flow of WSD process

Step 1: Translating input

Input can be given in both English and Telugu. If the given input is in English, then it is first translated into Telugu using the python package – googletrans. This package takes English sentences as input and translates then into desired target languages. For our use case, the target language is Telugu.

Step 2: Removing stop words

After the input is translated, it has to be processed before disambiguating the

words in the sentence. The initial step in this process is stop words removal.

Stop words are those words present in a language which add no semantic value

to the sentence. A list of Telugu stop words is generated and it is used to filter the translated sentence.

This is the first step in the pre-processing steps and also very important because presence of stop words may result in irregular results.

Step 3: Stemming

The next step, which is a part of pre-processing, is Stemming. It is the process of reducing a particular word to its root form. For example, the stem of the word *running* is *run.*

Telugu words are stemmed into their root from so that they can be given as an input to the pyiwn package.

Step 4: Applying Lesk

After all the pre-processing steps are performed, the resulting words are given as an input to the Lesk algorithm.

CHAPTER- 5

SYSTEM ANALYSIS

5.1 System Requirements

5.1.1 Pyiwn Python package

An Application Programming Interface (API) written in the Python language to provide access to WordNets not just in English but also for Indian languages. This API gives access to synsets, lexico-semantic relations between synsets, ontology nodes for different Indian languages i.e., Hindi, Kannada, Malayalam, Marathi, Sanskrit, Tamil, Telugu and Urdu etc.. In future, it will also provide access to speech data for words, glosses examples in Hindi WordNet. It only works with python version above 3.5.

5.1.2 Googletrans python package

Googletrans is a free and unlimited python library that implemented Google Translate API. This uses the Google Translate Ajax API to make calls to such methods as detect and translate. It's fast and reliable as it has very strong feature of auto language detection. The maximum character limit on a single text is 15000.

5.1.3 Flask

Flask is a lightweight WSGI web application framework. This means **flask** provides you with tools, libraries, and technologies that allow you to build a web application. It is

designed to get start, with the ability to scale up to complex applications quick and easy. It has become one of the most popular Python web application frameworks.

5.1.4 Python

Python Created by Guido van Rossum and first released in 1991, is an interpreted, high-level, general-purpose programming language. It is a scripted language as well as interpreted language. Python is an open source language.

It's feature contain:

- Object oriented
- Powerful
- Portable
- Contains large standard library
- Supports GUI Programming

There are mainly two versions of python i.e, 2 and 3. The current updated version of python is 3.8. This project works for version 3.7 and above.

5.1.5 Visual Studio Code

Visual Studio Code by Microsoft is a source-code editor; develop for Windows, Linux and macOS. Like other source code editors it also includes support for debugging, but the main intension of using this editor as it has embedded Git control and GitHub. It also includes syntax highlighting, intelligent code completion, snippets etc. It is highly customizable and is handy for the users as we can easily change the theme as wanted.It is used

in many organizations for development purposes by engineers. Also by being open source, anybody can use it without worrying about licenses.

CHAPTER-6

SYSTEM DESIGN

6.1 UML Diagrams Introduction:

UML, short for Unified Modelling Language is a standard language for specifying, visualizing, constructing, and documenting the elements of software systems. UML could be defined as a generic use case language used mainly for visual modelling used for mainly four purposes:

- Visualize
- Specify
- Construct, and
- Document

UML is usually used to model software systems but this is not a hard and fast rule. It is also used to model systems not related to software such as workflow in a manufacturing unit as an example. One thing to note is UML is not a programming language. But with the help of other software, code can be generated in different languages based on UML models. It has a direct relation with object oriented design and analysis.

6.2 Usecase Diagram:

A **use case diagram** can be defined as a representation of a user's interaction with the system, that shows the dependency between the user and the various use cases, in which the user is involved with the system. A use case diagram can identify the various kinds of users

of a system and also the different use cases and will generally be followed/accompanied by other types of UML diagrams as well.

6.2.2 Use case Diagram for WSD:

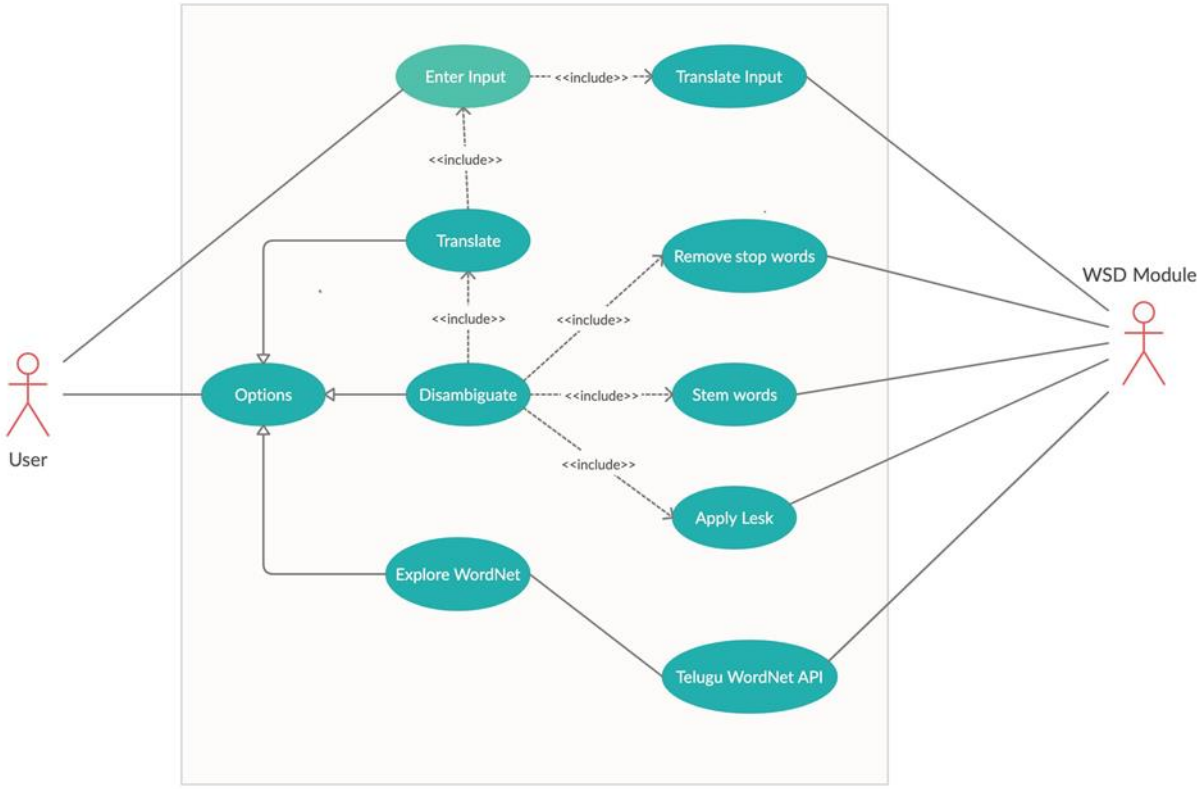


Figure 6.1 Usecase diagram for WSD

Description:

As the *Fig 6.1* the usecase diagram for WSD which consists of 2 actors:

User

WSD Module

User is displayed with a certain number of options where he needs to opt for an option.

The options provided are Disambiguate, Translate and Explore WordNet. The user then needs to enter the input on which he needs the output. WSD Module is the another actor where it takes care of the other activities where it translates the input given by the user. It has information regarding the telugu wordnet. The user can get information on that. During the disambiguation, it removes the stop words from the given input, applies lesk and even finds the stem words.

6.3 Class Diagram

6.3.1 Definition:

A **class diagram** in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

6.3.2 Class Diagram for WSD:

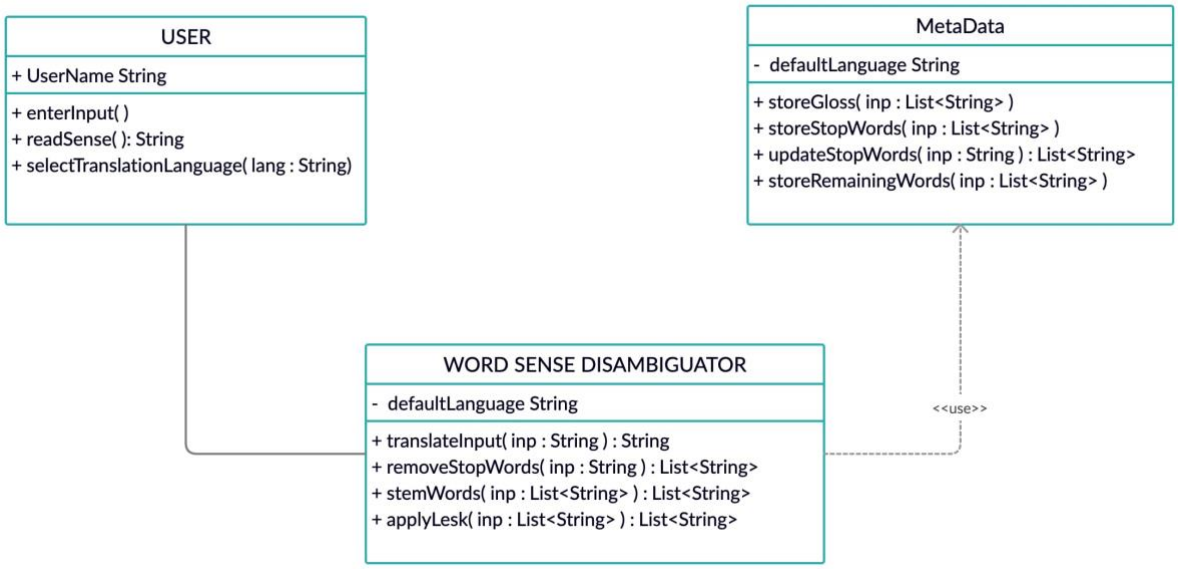


Figure 6.2 Class Diagram for WSD

Description:

The class diagram for WSD (*Fig 6.2)* consists of following classes.

1. User
2. Metadata
3. Word Sense Disambiguator

The User can open the WSD and enter the input he wants to

- enterInput() method of user allows the user to provide input and takes string as an input. This input is the one which is disambiguated by the system. The user can also enter input to deep dive into the 'pyiwn' interface
- readSense() is the function which, as the name suggests, is used to read the output provided by the WSD module. The output is in the form of key value pairs, where key

is the word in the sentence which is disambiguated and value is the best sense of the word.

- selectTranslationLanguage()method allows the user to select a language in which he/she wants the sentence to be translated into in the 'Translation' module. Based on the language given the output is translated into that language for a given sentence or word.

The Metadata has the following parameters with in it

- storeGloss() : Gloss is the meaning of a particular synset. It is a part of the object returned by the 'pyiwn' package when a word is given as an input. This is stored for all the words in a sentence as it is used in the algorithm.
- storeStopWords(): method stores a file which contains the stop words in Telugu. This file is generated by translating all stop words in English to Telugu. Each stop word is present in a new line to make it more readable by the script.
- updateStopWords(): method is used to update the file containing stop words as and when a new stop word is discovered.
- storeRemainingWords(): method stores the remaining words other than the word in question. These remaining words are uodated for every word in the sentence.

  This helps in disambiguating with the context.

Word Sense Disambiguator has the following parameters:

- translateInput(): method takes a string as an input and this method is responsible for the translation of the input provided by the user. This is necessary because it provides the user flexibility to provide input in any language, which is then converted into Telugu.

- removeStopWords(): method removes all the stop words present in the input given by the user. This step follows the translation step in the workflow.
- stemWords() : This method is used to convert words inot their root forms, i.e., stems the word. This is necessary to provide correct input to the disambiguator.
- applyLesk() method applies the lesk algorithm to the input given by the user and this algorithm disambiguates the input string.

6.4 Sequence Diagram:

6.4.1 Definition:

A Sequence diagram is an interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. A sequence diagram shows object interactions arranged in time sequence.

6.4.2 Sequence Diagram for WSD:

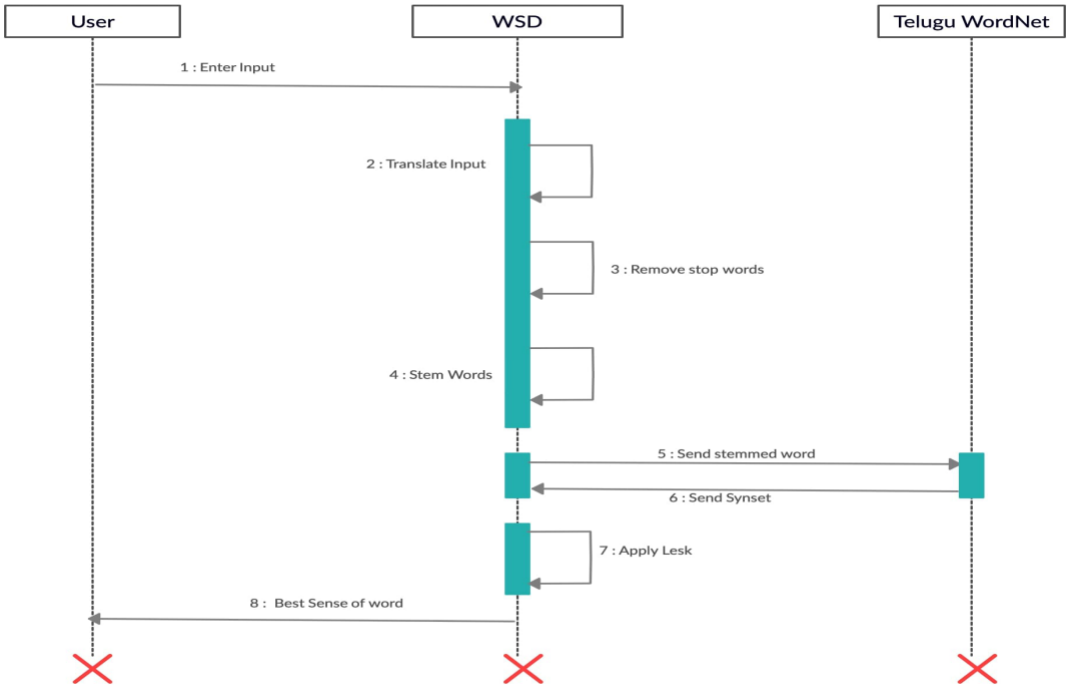


Figure 6.3 Sequence diagram for WSD

Description:

The user after opening the application, the welcome screen shows the home tab. Apart from that there are three tabs which show Disambiguate, IndoWordNet, Translate. The *Fig 6.3* shows the sequence of actions that take place. Then the user needs to enter the input. After entering the input, the input gets translated to Telegu using the google trans. Then the stop words are removed, the stop words are nothing but the frequently or commonly used English words for example, ‘is’, ’and’ are some of the stop words. The processed input after removing the stop words, the corresponding stem words are then appended to the input. The input to then sent to the

TeluguWordNet, the corresponding synset is sent to the WSD again. Then to the processed sentence or the input, the Lesk Algorithm is applied. Here the total sentence or the input is processed under the algorithm, the finest output or the Disambiguated output is shown to the user.

6.5 Activity Diagram:

6.5.1 Definition:

Activity diagrams can be defined as a visual depiction of workflows of activities which occur in a stepwise fashion. It also represents actions with options for conditional decision making, loops and concurrency. In UML, activity diagrams are meant to model both organizational and computational processes. Activity diagrams show the overall process and how control flows from start to end.

6.5.2 Activity Diagram for WSD:

The activity diagram for the WSD module (*Fig 6.4)* is depicted in the next page to avoid breaking it into two parts.

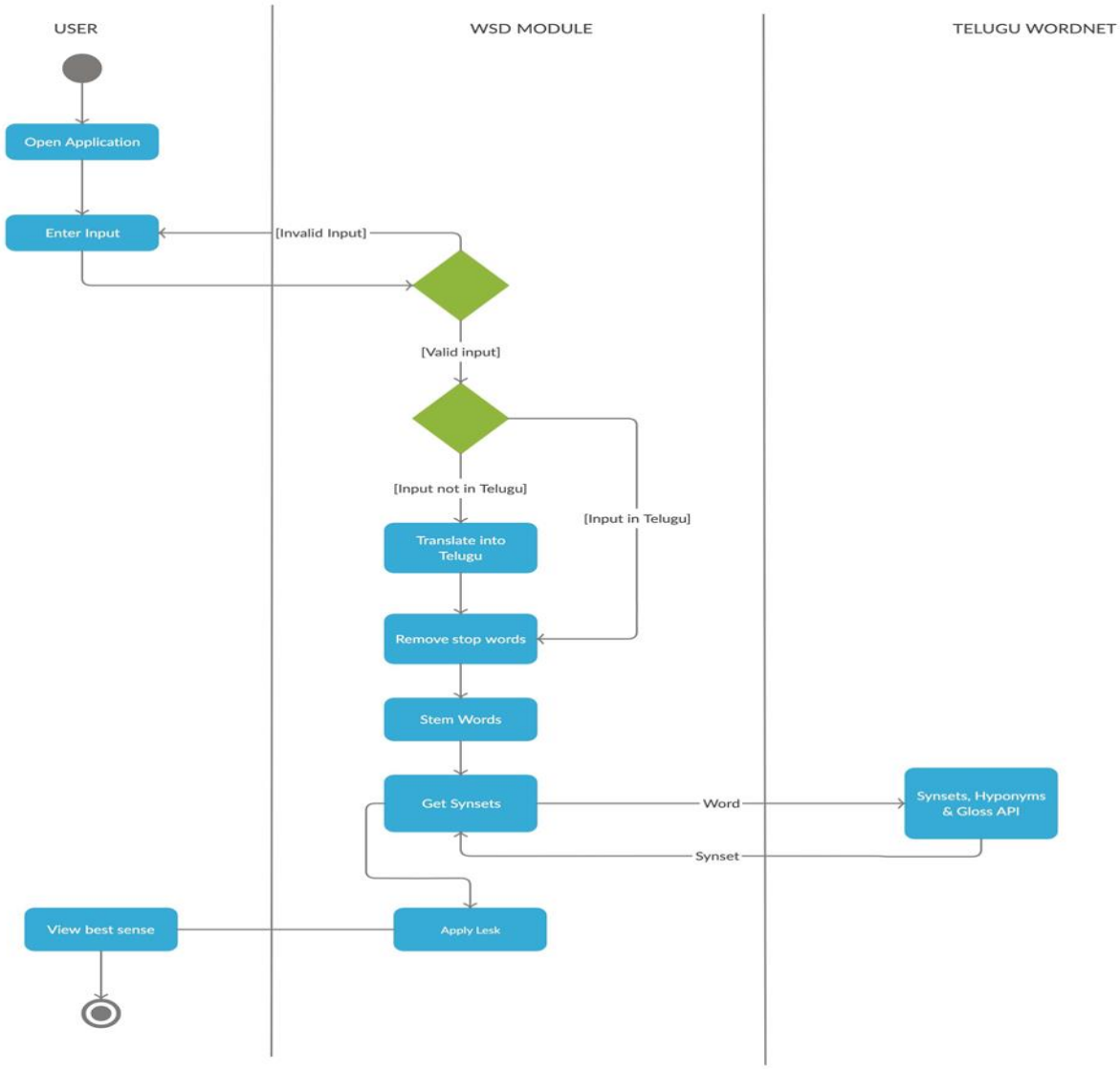


Figure 6.4 Activity diagram for WSD

Description:

The user begins the workflow by first opening the website and landing on the home page, on which there are multiple options such as – Disambiguate, Translate, IndoWordNet, etc. We will focus on the 'Disambiguate' option, as all other options are a part of it.

To begin with, the user enters an input sentence. The code then checks if it is valid input,i.e., if the input field is empty or not. If the input is present, then it is passed along to the 'Translate' module. Here, if the input is not in Telugu, it is translated to a Telegu sentence by using the 'googletrans' module. If the input is already a Telugu sentence then it is moved to the next stage, which is removing stop words.

In this stage, all the words which are not relevant to disambiguation or if they do not add any semantic value, they are eliminated from the sentence. This is known as stop word removal.

After this stage of pre-processing, the resultant set of words is passed on to the next step – Stemming. Here, the words are converted to their root form because for them to be a valid input to the IndoWordNet, they have to be in this format. Stemming is performed in Telugu and the result of this is a list of words which can now be given as an input to the IndoWordNet, which brings us to our next step – Getting Synsets.

In this step, the list of stemmed words is passed as input to the 'pyiwn' IndoWordNet package. It returns an object containing the synsets, hyponyms, POS tag, gloss, etc., of that word. These values are necessary to apply the Lesk algorithm. This is the last step performed by the WSD module.

Lesk is a Word Sense Disambiguation algorithm which uses the words around the word in question to determine the correct sense. Here is where the synsets, hyponyms, etc., from the

previous step are used. After applying this algorithm, the WSD module returns a map like structure back to the user containing a word and its best sense as a key-value pair. This is displayed in the UI using flask.

The three major swimlanes present in this diagram are User, WSD Module and Telugu Wordnet, which is a subset of the IndoWordNet provided by the ‘pyiwn’ package.

The above diagrams represent the flow of process in this project and gives an outline of the different functionalities provided by it.

CHAPTER-7

IMPLEMENTATION

7.1 CODING

7.1.1 Python code in WSD Application

This is the most important script in this project. This is where the disambiguation process happens. This can be seen as the brain of the project.

The algorithm applied in this code is Lesk. Comments before every function provide additional information about what that function is written for. The different functions and their purpose are:

- Translate_sentence(str) => translates the input sentence and returns a string of translated sentence
- get_Stop_words_list() => reads the stop_words.txt file and returns the word from that file as a list
- remove_stop_words(str) => filters the sentence passed and returns a list of filtered words
- get_sense(list) => the input to these functions is a list of words. Each word is provided as an input to the Lesk algorithm implemented in this method.
- disambiguate(str) => This is like a ‘main’ method of this script. All the functions mentioned above are called from this function in the proper order.

7.1.2 CODE:

```
"""
   Run this script as follows: python process.py
   It will output the result to files
   input_text.txt contains input words
   sans_stop_words.txt contains the input text after removing stop words
   synset.txt contains the output of matched synset
"""
from googletrans import Translator
import pyiwn

""" File initializations """
input_text=open('input_text.txt','w',encoding='utf-8') # Contains the initial input text
sans_stop_words=open('sans_stop_words.txt','w',encoding='utf-8') # Contains the input text after removing stop words
synset_output=open('synset.txt','w',encoding='utf-8') # Contains the output of matched synset

""" Sysnet initialization to Telugu"""
iwn = pyiwn.IndoWordNet(lang=pyiwn.Language.TELUGU)

""" Translate input into Telugu """
translator = Translator()
translated_input = translator.translate("sun is beautiful",dest='te') # Enter input in this line
```

```
""" Convert stop words as a list """
stop_words = []
with open('stop-words-tr.txt','r',encoding='utf-8') as stop_word:
    for line in stop_word:
        stop_words.extend(line.split())



""" Remove stop words """
processed_words = []
words = translated_input.text.split()
for word in words:
    input_text.write(word+'\n')
    if word not in stop_words:
        processed_words.append(word)
        sans_stop_words.write(word+'\n')


print(processed_words)
for word in processed_words:
    synset_output.write('\n'+str(iwn.synsets(word)))
```

7.1.3 Process of synset

7.1.4 Remove stopwords

```
"""
   Run this script as follows: python process.py
   It will output the result to files
   input_text.txt contains input words
   sans_stop_words.txt contains the input text after removing stop words
   final_output.txt contains the result
"""


import nltk
import pyiwn
from googletrans import Translator
from krovetzstemmer import Stemmer
from nltk import sent_tokenize, word_tokenize
from nltk.corpus import stopwords, wordnet
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.stem.snowball import SnowballStemmer
from nltk.tokenize import PunktSentenceTokenizer, sent_tokenize, word_tokenize


# nltk.download()


""" File initializations """
input_text=open('input_text.txt','w',encoding="utf-8") # Contains the initial input text
# print(input_text.readlines()[0])
```

```
sans_stop_words=open('sans_stop_words.txt','w',encoding="utf-8") # Contains the input text after removing stop words



""" Sysnet initialization to Telugu"""
iwn = pyiwn.IndoWordNet()
stemmer = Stemmer()

def print_line(file):
    file.write('\n')
    for i in range(10):
        file.write('------')

def stem_input(words):
    stems = []
    for w in words:
        stems.append(stemmer.stem(w))
    return stems

def translate_stems(stems):
    """ Translate input into Telugu """
    translated_words = []
    translator = Translator()
    for stem in stems:
```

```
        #print(translator.translate(stem,dest='te'))
        translated_words.append(translator.translate(str(stem),dest='te').text) # Enter input in this
line
    return translated_words

def translate_sentence(sentence):
    translator = Translator()
    return translator.translate(sentence,dest='te').text

def get_stop_words_list():
    """ Convert stop words as a list """
    stop_words = []
    with open('stop-words-tr.txt','r',encoding='utf-8') as stop_word:
        for line in stop_word:
            stop_words.extend(line.split())
    return stop_words

def remove_stop_words(translated_words):
    """ Remove stop words """
    processed_words = []
    stop_words = get_stop_words_list()
    words = translated_words
    for word in words:
        input_text.write(word+'\n')
```

```
        if word not in stop_words:
            processed_words.append(word)
            sans_stop_words.write(word+'\n')
    return processed_words


def get_sense(processed_words):
    rem_words = []
    rem_synsets = []
    rem_to_match = []
    matcher = []
    max_cnt = 0
    max_sense = []
    op = open("output.txt","a",encoding="utf-8")
    final_output = open("final_output.txt","w",encoding="utf-8")
    for word in processed_words:
        cnt = 0

        # Get Synsets of current word
        curr_synsets = iwn.synsets(word,pos=pyiwn.PosTag.NOUN)

        # Get the remaining words and their synsets into rem_synsets
        for rem_word in processed_words:
            if rem_word != word:
                rem_words.append(rem_word)
```

```
for rem_word in rem_words:
    rem_synsets = (iwn.synsets(rem_word,pos=pyiwn.PosTag.NOUN))
op.write('\n'+str(rem_synsets))
print_line(op)

# For each synset of remaining words,
# generate a list of words from the meanings of the synset and its hyponyms
# o/p : rem_to_match - list of words from the meanings of the synset and its hyponyms
for synset in rem_synsets:
    hyponyms = iwn.synset_relation(synset, pyiwn.SynsetRelations.HYPONYMY)
    for hyponym in hyponyms:
        rem_to_match.extend(hyponym.gloss().split())
    rem_to_match.extend(synset.gloss().split())
op.write('\n'+str(rem_to_match))
print_line(op)

# For each synset of current word,
# match the words from its meaning to all the words in rem_to_match
# If count matches increment it and if it is maximum, it is taken as the sense
for synset in curr_synsets:
    matcher = []
    cnt = 0
    hyponyms = iwn.synset_relation(synset, pyiwn.SynsetRelations.HYPONYMY)
    for hyponym in hyponyms:
```

```
            matcher.extend(hyponym.gloss().split())
        matcher.extend(synset.gloss().split())
        op.write('\n'+str(matcher))
        print_line(op)
        for matcher_word in matcher:
            if matcher_word in rem_to_match:
                cnt += 1
        if cnt >= max_cnt:
            max_cnt = cnt
            max_sense = synset
    final_output.write(word + '=>' + max_sense.gloss() + ' : '+str(max_cnt)+'\n')



if __name__ == '__main__':

    #words = word_tokenize("saree shop")
    words = word_tokenize("grip on subject")
    #words = word_tokenize("mounika stands first in class")
    #words = word_tokenize("silk saree is beautiful")
    stems = stem_input(words)
    translated_words = translate_stems(stems)
    processed_words = remove_stop_words(translated_words)
    get_sense(processed_words)
```

7.1.5 Translate

```
from googletrans import Translator
translator = Translator()
s = translator.translate("stand",dest='te')
print(s.text)
```

7.1.6 Build stopwords

```
"""
   Run this script as follows: python build_stop_words.py <stop word to add>
   Ex:-  python build_stop_words.py inside
   will add the telugu translation of "inside" to stop-words-tr.txt
"""
import sys
from googletrans import Translator

""" Translate stop word into Telugu """
translator = Translator()
s = translator.translate(sys.argv[1],dest='te')

""" Append stop word to file """
f = open('stop-words-tr.txt','a')
```

```
f.write('\n'+s.text)


""" Translated a file containing English stop words to Telugu """
with open('stop-words-en.txt','r',encoding='UTF8') as sw:
    with open('stop-words-tr.txt','w',encoding='UTF8') as n:
        for line in sw:
            trans = translator.translate(line,dest='te')
            n.write(trans.text+'\n')
```

7.2 IMPLEMENTATION FLOW

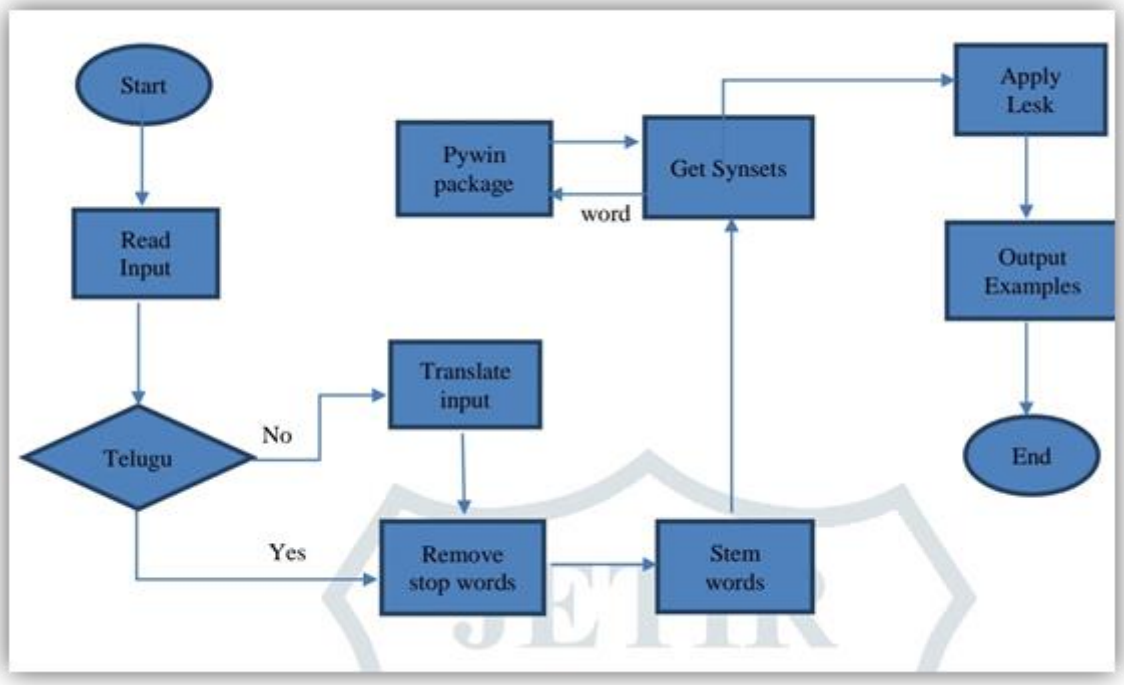


Fig: 7.2 flow diagram of WSD

- In Implementation different steps involved are Reading Input, Translating Input, removing stop words, getting synsets, Applying Lesk, Displaying results (examples).

- To achieve this knowledge-based algorithm is used which includes finding the overlap between the features of different senses of ambiguous word and the features of the word in its context. That is performed by taking words in the neighborhood of the target word and the sense, which has the maximum overlap, is selected as the context appropriate sense. Semantic distinguishing in Telugu is achieved after performing necessary pre-processing steps such as translating the words, stop words filtration, stemming, etc. In the below input sentences the word grip and silk have same translation ‘pattu’ in telugu but the meaning according to the context of sentence is different. So, in the output examples are given for each word in the context of the sentence.

CHAPTER-8

OUTPUT SCREENSHOTS

8.1 Example (one) screenshots: Silk saree is beautiful

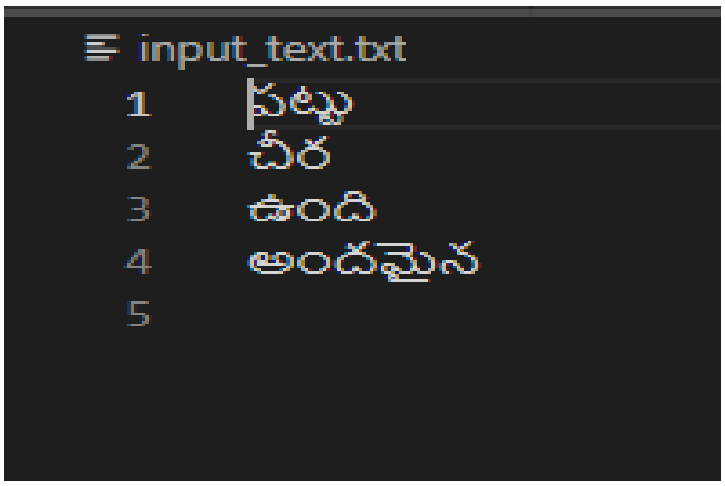


Fig:8.1 Input in telugu

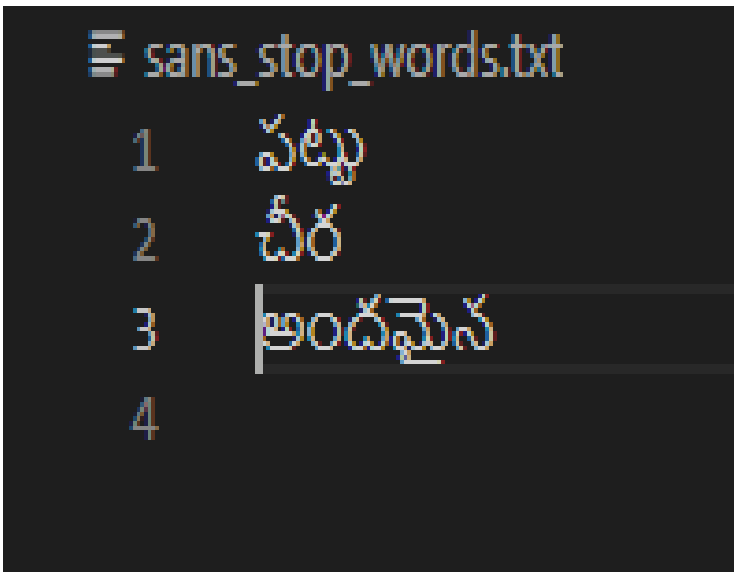


Fig:8.2 Input after removing stop words

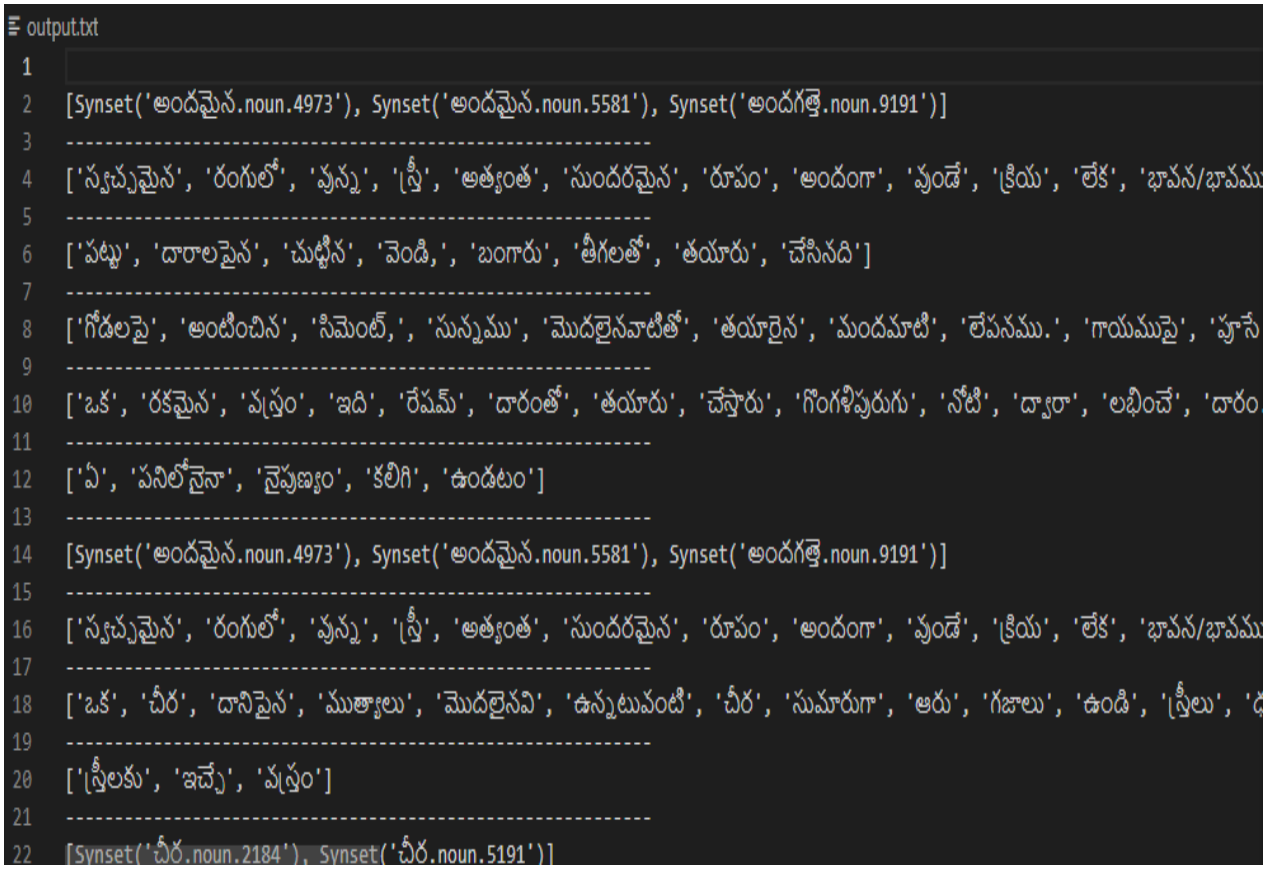


Fig: 8.3Matching synset

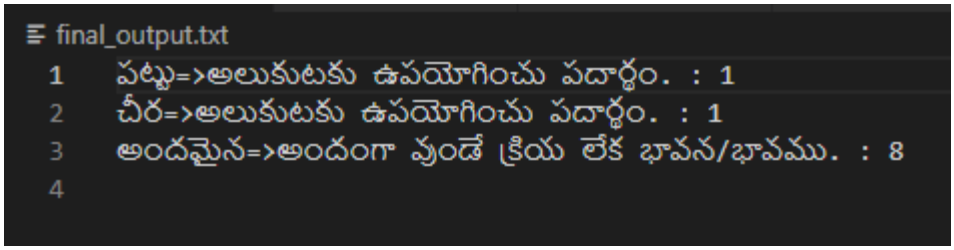


Fig:8.4 Final output examples

8.2 Example (second) Screenshots: Grip on subject

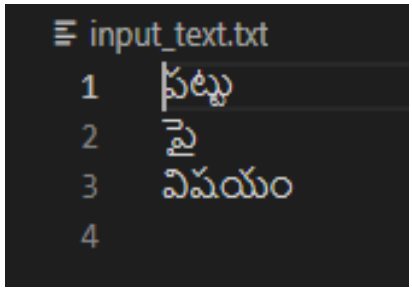


Fig: 8.5 Input in telugu

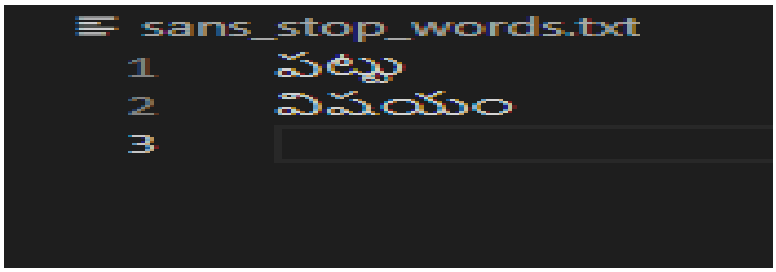


Fig::8.6 Input after removing stop words

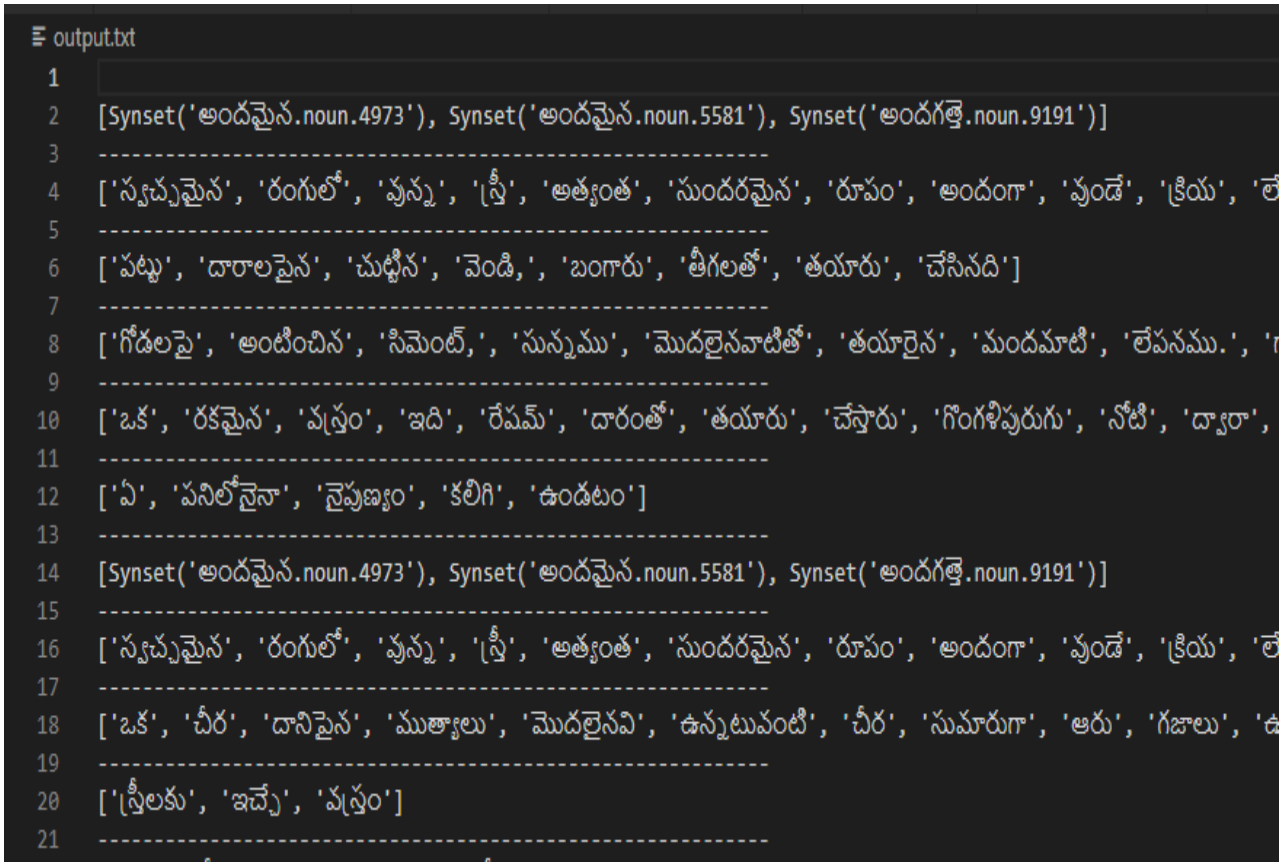


Fig:8.7 Matching synset

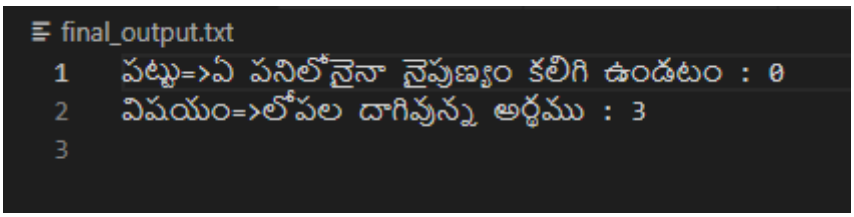


Fig:8.8 Final output examples

CHAPTER-9

TESTING

Software Testing, as the name suggests, refers to the process of testing if the expected values match the actual values produced by the system. It is performed to ensure that the system works as expected and without any defects. It is performed by testing various parts of a system by giving different kinds of input to cover all bases.

If the tests all pass, then we can say that the system works as expected and is ready to for production purposes. Generally, testing is not performed by the same people that develop the application because, people who develop it will not wish to perform tests which may cause errors or breakage.

9.1 TYPES OF TESTING:

There are various of kinds of testing methods such as:

- Unit testing
- Functional Testing
- Integration Testing
- Acceptance Testing, and so on.

In this section we perform Unit and functional Testing.

9.1.1 Unit Testing:

Unit testing is the process of testing individual units of a system. Or testing the smallest part of a system. It involves providing inputs to specifically test the individual functions or methods rather than the entire application.

For example, in this application, individual elements or functions include – translator, stop word remover, stemmer, different elements of Lesk, etc.

To test the various outputs of these functional units to various inputs, unit testing is performed. It involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integrating.

9.1.2 Functional Testing:

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

- Valid Input: Clear and Meaningful query with keywords.
- Invalid Input: Incomplete and improper query with misspells and missing keywords
- Functions: Identified functions must be exercised.
- Output: Correct identification of response and accurate retrieval of messages, based on intents matched.

9.2 RESULTS:

Test Case No.	Test Case	Expected Output	Actual Output	Result
1	Query in English language	The sentence is disambiguated and correct sense of each word is displayed	The sentence is disambiguated and correct sense of each word is displayed	**PASS**
2	Query in Telugu language	The sentence is disambiguated. Since its already in Telugu, no translation is applied.	Since the input is already in Telugu, translator is not called and the sentence is disambiguated	**PASS**
3	NULL String as input	No output is given as input is NULL	There is no output since the input field cannot be empty. So, it matches expected output	**PASS**
4	Special characters (like !, @, #, \, etc.) as input	The special characters filtered from the input before further processing	The special characters are filtered from the input before further processing	**PASS**
5	Query containing only stop words and special symbols	No output since all stop words and special characters will be filtered	No output since all stop words and special characters will be filtered	**PASS**

6	Input contains only one word	The application performs disambiguation but the result may not always be right.	Disambiguation is performed but the result may not always be right since there are no neighboring words	**PASS**
7	Input query has incorrectly spelt words	No output, since the translation will fail.	No output, since the translation may fail or the WordNet may not contain that word	**PASS**

Table 9.1: Result of Test Cases

CHAPTER-10

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion:

This paper mainly focuses on what Word Sense Disambiguation exactly is, what were the methods involved in solving them. Until now many approaches were made in solving them which mainly focused on English language and other few regional languages where the approaches included were both supervised and unsupervised techniques. So, this paper focuses on WSD for Telegu which is an Indian regional language spoken by the people of states of Telangana and Andhra Pradesh. This paper implements Lesk on the sentence and succeeded pretty well in its sense where the output totally depended on the synset. Word Sense Disambiguation for Telugu is achieved after performing necessary preprocessing steps such as translating the words, stop-words filtration, stemming, etc., and later implementing the Lesk algorithm on the processed words. This works for single sentences where the input is given in a sentence form.

As the number of words used increases, the complexity also increases and the number of synsets to process also increases. There is a scope to extend WSD for a whole paragraph or a whole document.

10.1 Future Enhancement

For now, the Word Sense Disambiguation process works best for single sentences up to 10-12 words at a time. As the number of words increases, the translator may not be able to gauge the meaning of the sentence and may give wrong sentences as output.

As a future scope for this project, we would like to extend this Word Sense Disambiguation process for longer sentences, paragraphs or documents. The limitation posed by the translator's ability to translate the entire document has to be overcome to make this plausible. Also, as the number of words to process increases, the complexity of the algorithm also increases, as for every word it has to generate the synsets of all other words in that sentence. This is the second future enhancement that can be applied by improving the efficiency of the algorithm to handle large documents or sentences.

CHAPTER-11
REFERENCES

[1] A. Brahmananda Reddy "Integrated Feature Selection Methods for Text Document Clustering", Integrated Feature Selection Methods for Text Document Clustering, 2015

[2] A. Brahmananda Reddy, A.Govardan "Ontology for an Education System and Ontology Based Clustering", The Fifth International Conference on Fuzzy and Neuro Computing (FANCCO - 2015), IDRBT, Hyderabad, 2015

[3] J.Sreedhar , Dr.S.Viswanadha Raju, Dr.A.Vinaya Babu "A Study of Critical Approaches in WSD for Telugu Language Nouns: Current State of the Art" International Journal of Scientific &Engineering Research, Volume 5, Issue 6, ISSN 2229-5518, June 2014

[4] https://www.simplilearn.com/feasibility-study-article

[5] Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[6] George A. Miller (1995). WordNet: A Lexical Database for English Communications of the ACM Vol. 38, No. 11: 39-41.

[7] Pushpak Bhattacharyya "IndoWordNet" IIT Bombay, March 2010

[8] Ritesh Panjwani, Diptesh Kanojia, Pushpak Bhattacharyya "pyiwn: A Python-based API to access Indian Language WordNets" IIT Bombay, 2018

© 2022 JETIR June 2022, Volume 9, Issue 6
www.jetir.org (ISSN-2349-5162)

JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Semantic Distinguishing To Identify The Context Of Words In Telugu Sentences

[1]K. Neha, [2]P. Shivani, [3]M. Indu, [4]S. Arjun
[5]Dr.M.V.Vijaya Saradhi
[1,2,3,4] B.Tech (IV-CSE), Department of Computer Science and Engineering
[5]Head of the Department (HOD), Department of Computer Science and Engineering, ACE Engineering College, Ghatkesar, Hyderabad, Telangana, India

Abstract : *Natural Language Processing domain is mainly used in fields such as Information retrieval and Text mining. These do not have a built-in semantic disambiguate and thus, it has to be performed explicitly. This has been largely performed on sentences in the English language to identify the context of a word in that sentence. This project's main idea is to implement semantic distinguishing in sentence by giving examples for regional language Telugu. Since, Telugu vocabulary is huge, occurrences of ambiguity is common and this project is aimed at resolving them. To achieve this, a synset is used, which is an interface for the Telugu WordNet. The algorithm used in this project is the Lesk algorithm. It is based on the assumption that the correct meaning or sense of a word is dependent upon the words present in its neighborhood. More specifically, context of a word is determined by comparing the meaning of each sense of the word with the meanings of the neighboring words. This project aims to get the examples for each ambiguous word in the sentence using Lesk algorithm.*

IndexTerms **- Semantic distinguishing , Telugu, Lesk algorithm, WordNet, Python.**

I. Introduction

In Natural Language Processing, Word Sense Disambiguation (WSD) is the problem of determining which meaning of a word is activated by the use of the word in a particular context, a process which appears to be largely unconscious in people. WSD is a natural classification problem: Given a word and its possible senses, as defined by a dictionary, classify an occurrence of the word in context into one or more of its sense classes. The features of the context (such as neighboring words) provide the evidence for classification.

A famous example is to determine the sense of pen in the following passage: Little John was looking for his toy box. Finally he found it. The box was in the pen. John was very happy.

1. WordNet lists five senses for the word pen: pen — a writing implement with a point from which ink flows.
2. Pen — an enclosure for confining livestock.

3. Playpen, pen — a portable enclosure in which babies may be left to play.
4. Penitentiary, pen — a correctional institution for those convicted of major crimes.
5. Pen — female swan.

Semantic Distinguishing is the task that automatically identifies the exact sense of the ambiguous word based on the context in which the word is used and give examples in the same context to understand the sense of the sentence. The proposed system aims to perform semantic distinguishing for Telugu .The input is taken in English and translated to Telugu using googletrans in python. Then required pre-processing steps are performed on the sentence. Stop words are removed, Stemming is performed, Lesk algorithm is applied and various examples are given as output.

II. LITRATURE SURVEY

The research papers and their brief summaries are as follows:

1. J.Sreedhar, in their paper spoke about the various methods which can be used to perform Word Sense Disambiguation for Telugu. The paper began by first talking about the taxonomy of Word Sense Disambiguation in NLP and explained in brief about each of them. In their own words, Word Sense Disambiguation is defined as follows: “Word Sense Disambiguation (WSD) is the ability to computationally determine which sense of a word is activated by its use in a particular context.” In the following sections of the paper, current state of the art methods used for Word Sense Disambiguation are briefly explained. Some of the methods discussed in the paper are: - Method proposed by Walker, in which a thesaurus is used. Each word is assigned a particular subject categories in the thesaurus. Each subject assigned to a particular word is assumed to be a particular sense and WSD is performed based on this assumption. - Method proposed by Quillian in mid 1960s in which he proposed the use of semantic network representation of a machine readable dictionary. In this, a node represents the meaning and this node is used to connect words. Many such methods have been specified in this paper along with a brief description of each method. Lesk is one of the proposed method which uses a WordNet. Improvements in this project: Even though the paper discusses various methods for Word Sense Disambiguation, it does not provide any implementation details about it. Lesk has been chosen to perform WSD in this project, as it has not been researched extensively for Telugu, unlike other algorithms. Many methods were analysed in this paper and provided a concise explanation. Since no implementation details were provided, this project aims at implementing Lesk algorithm described in the paper.

2. Ritesh Panjwani , in their paper spoke about their python-based Application Programming Interface (API) for Indian WordNets. This can be used as a module by getting it from pypi.org, where it is present as an open source project. In their own words, the aim of their project: “With our work, we aim to provide an accessible, robust, easy-to-use API for Indian language WordNets. The paper starts off by talking related work that already exists, such as ‘The Java WordNet Library’, which is a Java API to access the WordNet. Their motivation was to create an API for IndoWordNet which was python based so that it could be easily integrated with projects.

3. Word Sense Disambiguation for Telugu Using Lesk Sudheendra Poluru, A.Brahmananda Reddy, Rahul Manne, Lokesh Bathula & Nikhilender B. Reddy .

III. EXISTING SYSTEM

At present almost all word sense disambiguation focus on English and English synsets. It cannot work for the regional languages. It would be a great problem to find the ambiguity for the regional languages. Theme or context finding is an important part in processes such as page ranking and very few systems exist that perform word sense disambiguation in Telugu.

The drawbacks of the current system include the following:

Mostly applicable to English language: Most of the work on Word Sense Disambiguation is being performed in the English language. All new algorithms are first applied to the English language and only then, they are adapted to other languages slowly. The stemming required to perform Word Sense Disambiguation is performed by many python packages or similar only for English. There are no open source stemmers for Indian regional languages, for example, as a python module. There is no interface to the IndoWordNet provided by the 'Pyiwn' package which can be used to view the different synsets of a word, its hyponyms, gloss, etc.

IV. PROPOSED SYSTEM

Our solutions aims at performing word sense disambiguation on Telugu sentences to help gauge the theme of the sentence. It gives example for each word in the searched sentence in Telugu with same sense (meaning in the sentence).

The proposed system aims to perform Semantic distinguishing for regional Telugu in the following ways:

1. The input is taken in English and translated to Telugu using googletrans package in the python.
2. Pre-processing steps are performed on the sentence like removing stop words, Stemming.
3. Pyiwn python package is used, which provides an API to access the IndoWordNet.
4. The relevant synsets are used and provided as input to the Lesk algorithm.
5. Lastly, it outputs example for each word in the searched sentence in Telugu with same sense (meaning in the sentence).

LESK ALGORITHM

The Lesk algorithm is a classical word sense disambiguation algorithm introduced by Michael E. Lesk. It is based on the assumption that the sense of a word is dependent upon the words present in its neighborhood. More specifically, sense or context of a word is determined by comparing the meaning of each sense of the word with the meanings of the neighboring words. This comparison is made by matching the words in meanings and the sense whose meaning matches the maximum number of times is taken as the resultant sense of that particular word in a given sentence.

Steps in Lesk Algorithm:

1. Consider a word for processing.
2. Next, take the remaining words and create a list of words, which consist of the gloss and examples of those words. Let us call it 'matcher_list'.

3. Remove stop words from these lists to reduce irregular matches.
4. Now for every synset of the word to be processed, create a similar list of words containing the gloss and examples of that synset. Let us call this list 'synset_list'.
5. Match the above two list to find overlap of words.
6. The synset_list, which has the maximum overlap with the matcher_list, is taken as the best synset and the gloss of the synset is taken as the best sense.

V. IMPLEMENTATION

In Implementation different steps involved are Reading Input, Translating Input, Removing stop words, Getting synsets, Applying Lesk, Displaying results (examples).

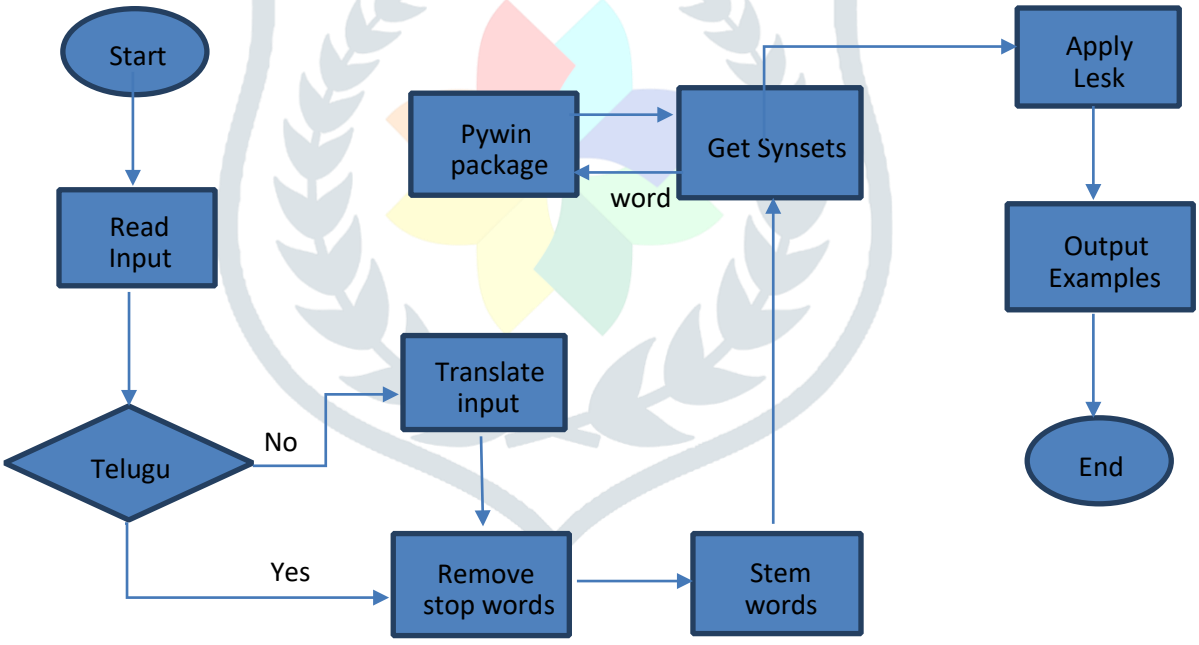


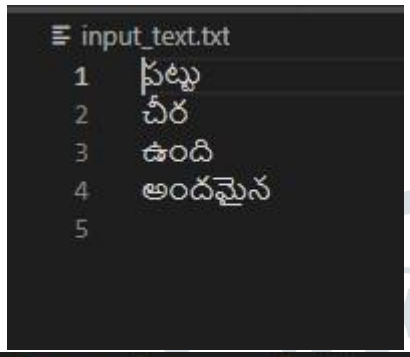
Fig 1: Implementation flow

To achieve this knowledge based algorithm is used which includes finding the overlap between the features of different senses of ambiguous word and the features of the word in its context. That is performed by taking words in the neighborhood of the target word and the sense, which has the maximum overlap, is selected as the context appropriate sense. Semantic distinguishing in Telugu is achieved after performing necessary pre-processing steps such as translating the words, stop words filtration, stemming, etc. In the below input sentences the word grip and silk have same

translation ‘pattu’ in telugu but the meaning according to the context of sentence is different . So, In the output examples are given for each word in the context of the sentence.

Example 1:

Input text : Silk saree is beautiful



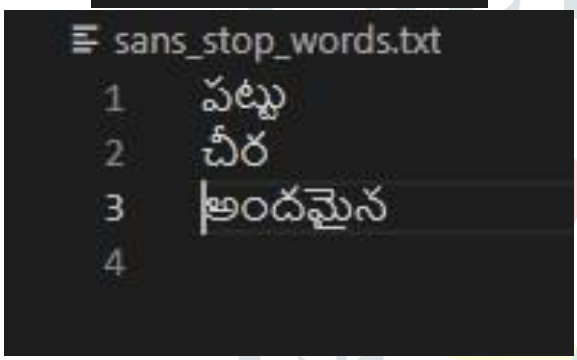


Fig 2.1: Input in telugu

Fig 2.2: Input after removing stop words

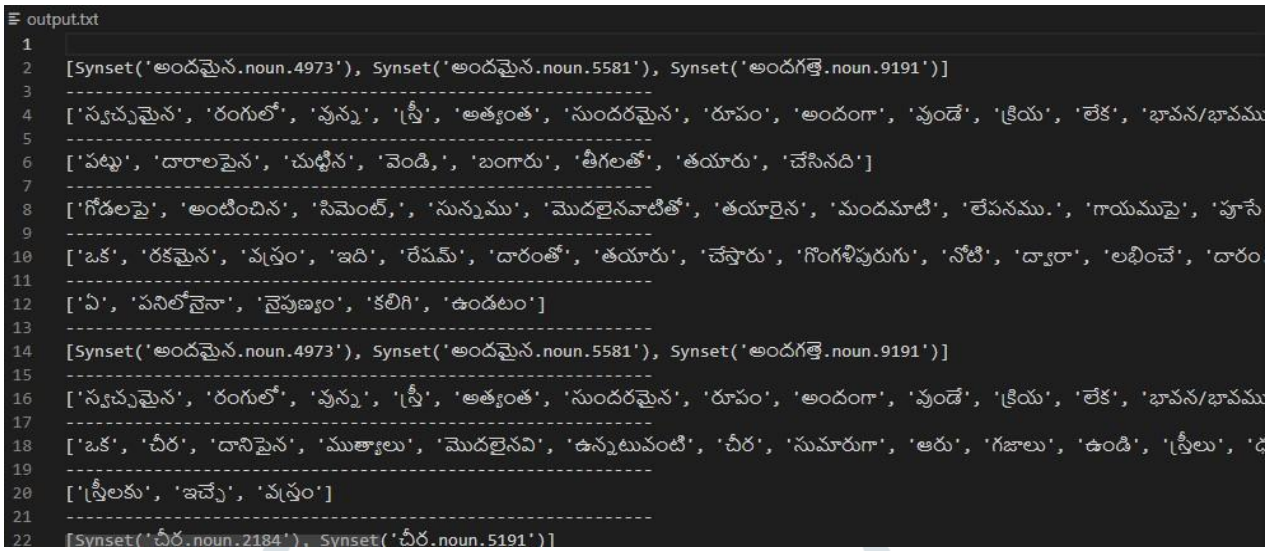


Fig 2.3: Matching synset

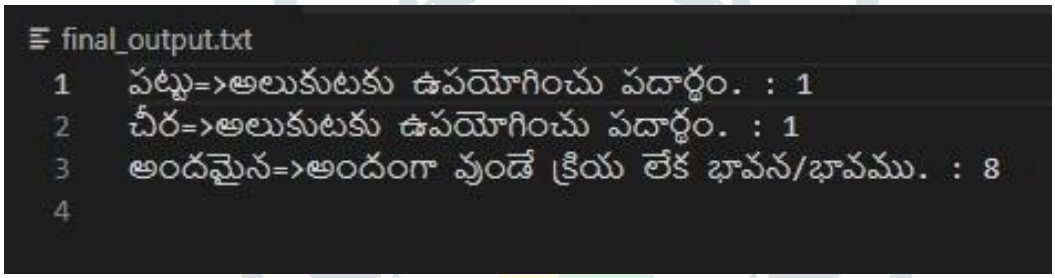
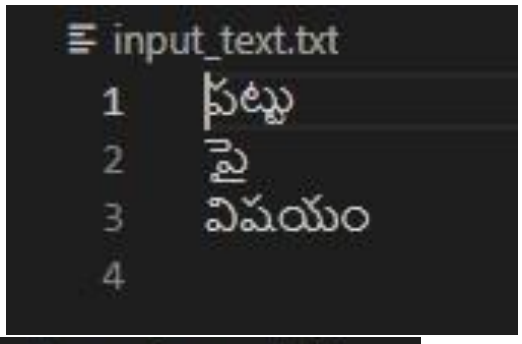


Fig 2.4: Final output examples

Example 2:

Input text : Grip on subject



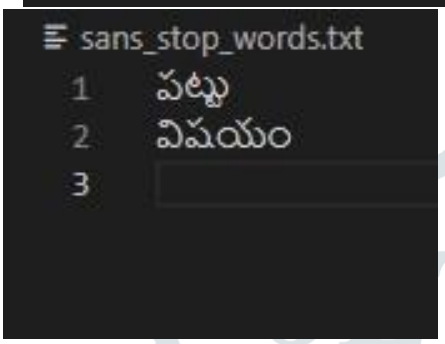


Fig 3.1: Input in telugu

Fig 3.2: Input after removing stop words

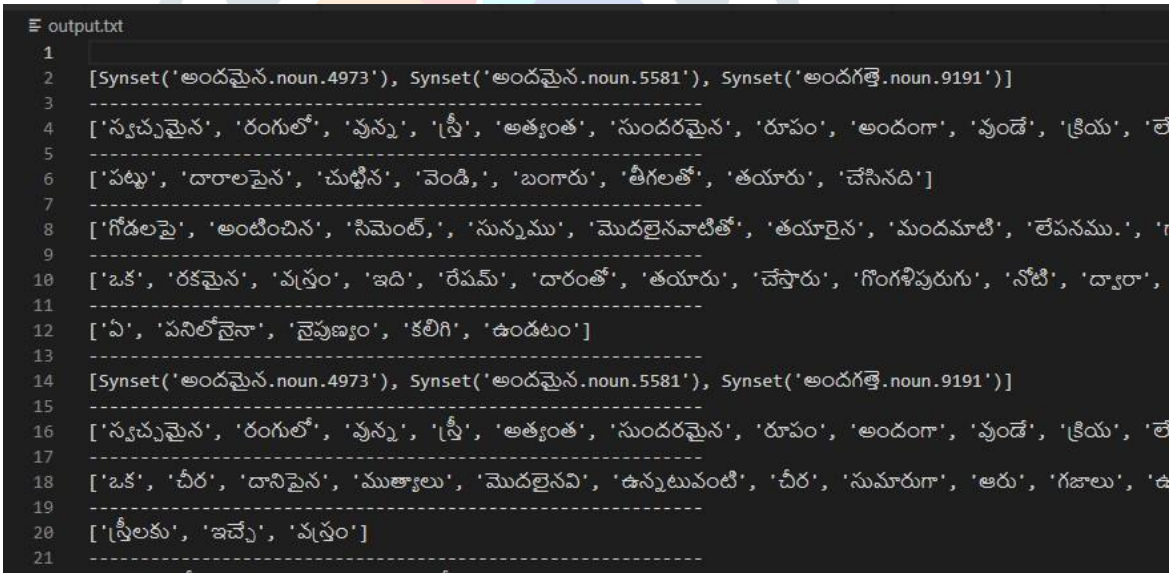


Fig: Matching synset

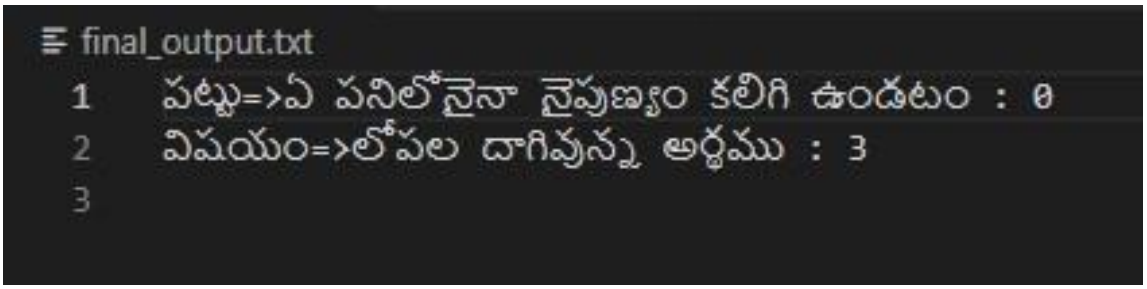


Fig : Final output examples

VI. CONCLUSION

This paper mainly focuses on what Semantic Disambiguation. Many approaches were made in solving them, which mainly focused on English language. So, this paper focuses on Semantic distinguishing for Telugu which is an Indian regional language spoken by the people of states of Telangana and Andhra Pradesh. Semantic distinguishing to identify the context of a word in Telugu sentences is achieved after performing necessary preprocessing steps such as translating the words, stop-words filtration, stemming, etc., and later implementing the Lesk algorithm on the processed words and outputs examples to understand the context of the sentences. This works for the input, which is given in single sentences form.

VII. Acknowledgment

We would like to thanks to our guide Dr.M.V.Vijaya Saradhi, Head of the Department of Computer Science and Engineering, ACE Engineering College and Mrs. Soppal Kavitha for their continuous support and guidance. Due to their guidance, we could complete our work successfully.

References

[1] A. Brahmananda Reddy “Integrated Feature Selection Methods for Text Document Clustering”, Integrated Feature Selection Methods for Text Document Clustering, 2015.

[2] A. Brahmananda Reddy, A.Govardan “Ontology for an Education System and Ontology Based Clustering”, The Fifth International Conference on Fuzzy and Neuro Computing (FANCCO - 2015), IDRBT, Hyderabad, 2015.

[3] J.Sreedhar , Dr.S.Viswanadha Raju, Dr.A.Vinaya Babu “A Study of Critical Approaches in WSD for Telugu Language Nouns: Current State of the Art” International Journal of Scientific &Engineering Research, Volume 5, Issue 6, ISSN 22295518, June 2014.

[4] Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[5] George A. Miller (1995). WordNet: A Lexical Database for English Communications of the ACM Vol. 38, No. 11: 39-41.

[6] Ritesh Panjwani, Diptesh Kanojia, Pushpak Bhattacharyya “pyiwn: A Python based API to access Indian Language WordNets” IIT Bombay, 2018.