A MINI PROJECT REPORT ON

"POS Tagging and Translation for Multiple Languages"

SUBJECT - Lab Practice V (Natural Language Processing)

SUBMITTED BY

Ms. Samruddhi Sandip Kangude

Under the Guidance of

Prof. Bharati P. Khond.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

ALARD COLLEGE OF ENGINEERING AND MANAGEMENT

MARUNJI, PUNE

FINAL YEAR ENGINEERING

(Academic Year: 2024-2025)

Alard Charitable Trust's

ALARD CHARITABLE TRUST'S

ALARD COLLEGE OF ENGINEERING AND

MANAGEMENTMARUNJI, PUNE

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINELEARNING

CERTIFICATE

This is certified to that **Ms. Samruddhi Sandip Kangude (72252518E)** has completed the Mini -Project work and prepared report on "**POS Tagging and Translation for Multiple Languages**" In satisfactory manner as a partial fulfilment on requirement of **Final Year** Engineering in **AIML** for academic year 2024-25.

Prof. Bharati P. Khond (Project Guide)

Prof. Disha Nagpure (HOD)

Abstract

In the field of Natural Language Processing (NLP), multilingual support is crucial for enhancing user accessibility and linguistic inclusivity. This project presents a Python-based software system for **Part-of-Speech (POS) tagging and language translation** across multiple Indian languages including Hindi, Marathi, Urdu, Tamil, Bengali, Gujarati, Kannada, and Telugu. The system utilizes the langdetect library for automatic language identification and the googletrans API for translation. For syntactic analysis, the nltk library is employed to perform POS tagging on both original and translated texts. The tool provides real-time processing by taking English input, translating it into multiple Indian languages, and tagging each translated sentence to identify parts of speech.

The system is entirely **software-based**, designed for general-purpose computing platforms. It requires minimal **hardware** resources, such as a standard computer with internet access, making it suitable for integration into educational, linguistic research, and NLP-based applications. The application showcases the capabilities of Python in developing robust and scalable linguistic tools for low-resource language processing.

Keywords: Natural Language Processing (NLP), POS Tagging, Translation, Language Detection, Multilingual Processing, Text Tokenization, Syntactic Analysis, Indian Languages NLP.

Software Used: Python 3.x – Programming Language, nltk (Natural Language Toolkit) – For POS tagging and text processing, Jupyter Notebook – For development.

Hardware Used: Processor- Dual Core or higher, RAM- 4GB or more, Storage: 8GB RAM

Acknowledgments

It gives me great pleasure to present the preliminary project report on 'POS Tagging and Translation for Multiple Languages'. I would like to take this opportunity to express my sincere gratitude to my internal guide, Prof. Bharati P. Khond Madam, for providing me with all the help and guidance I needed throughout the course of this project. I am truly thankful for her valuable suggestions, constant encouragement, and kind support. I am also grateful to Prof. Disha Nagpure Madam, Head of the AIML Department, Alard College of Engineering and Management, Pune, for her indispensable support and thoughtful suggestions that significantly contributed to the progress of my project. Lastly, I extend my heartfelt thanks to all the staff members of the AIML department and the librarian for providing essential resources such as access to the laboratory, required software platforms, and uninterrupted internet connectivity, which were crucial for the successful development of my project.

Ms. Samruddhi Sandip Kangude (72252518E)

INDEX

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In today's linguistically diverse and globally connected world, the ability to understand and process multiple languages is a vital aspect of Natural Language Processing (NLP). Core tasks like Part-of-Speech (POS) tagging and language translation play a key role in enabling intelligent systems to analyse, interpret, and generate human language across various contexts. While POS tagging assigns grammatical categories such as nouns, verbs, and adjectives to words in a sentence, translation bridges communication gaps by converting text from one language to another, enhancing accessibility and inclusivity.

This project aims to develop a Python-based system that leverages open-source libraries to perform multilingual text processing. It uses the langdetect library to detect the language of the input, googletrans API to translate English text into several Indian languages (like Hindi, Marathi, Tamil, Bengali, etc.), and the nltk library to execute POS tagging on the translated sentences. By integrating these components, the system showcases a lightweight, yet powerful tool suitable for both educational and real-world NLP applications.

1.2 MOTIVATION

In a linguistically diverse country like India, communication across various languages presents a significant challenge, particularly in the digital domain. Most existing language processing tools cater primarily to English or a few globally dominant languages, resulting in limited support for many Indian regional languages. This technological gap motivated me to develop a system capable of bridging these linguistic barriers by implementing Part-of-Speech (POS) tagging and translation for multiple Indian languages, making language technologies more inclusive and accessible.

The rising demand for multilingual Natural Language Processing (NLP) in areas such as education, social media, e-governance, and customer service further inspired this project. By combining language detection, translation, and syntactic analysis into a single platform, the project aims to support both practical use cases and linguistic research. Moreover, working with robust Python libraries like NLTK, langdetect, and googletrans offered a valuable opportunity to apply theoretical knowledge in real-world scenarios, encouraging the development of meaningful, cross-language AI solutions.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Survey

POS Tagging and Translation for Multiple Languages has become a critical area of research in Natural Language Processing (NLP), particularly in multilingual contexts. Early systems were rule-based and language-specific, making them difficult to scale. However, with the advent of statistical methods and machine learning, more robust and adaptable models have been developed. More recently, neural network-based approaches and APIs like Google Translate have made multilingual translation and syntactic analysis more accessible.

Overview of Research in POS Tagging and Language Translation Across Multiple Languages

Study 1: A 2000 study by Brill and Moore introduced a rule-based POS tagger for English text that leveraged transformation-based error-driven learning. Although effective, the system lacked scalability across multiple languages due to dependency on handcrafted rules.

Study 2: A 2011 paper by Petrov et al. presented a universal POS tagset that facilitated consistent tagging across multiple languages. This laid the foundation for multilingual POS tagging using machine learning approaches like Conditional Random Fields (CRFs) and Hidden Markov Models (HMMs).

Study 3: A 2017 study by Plank et al. explored neural POS tagging across low-resource languages using bidirectional LSTM (BiLSTM) models. They demonstrated that deep learning models could achieve high accuracy even with limited annotated data by leveraging multilingual embeddings.

Study 4: A 2020 paper discussed the integration of Google Translate API with NLTK for processing multilingual text. This study emphasized the importance of translation quality and tokenization in preserving syntactic structure during POS tagging.

Study 5: In a recent 2022 study, multilingual POS tagging systems were evaluated on Indian languages like Hindi, Marathi, and Tamil. The research used pre-trained language models (like BERT multilingual) and showed significantly better performance compared to traditional methods.

CHAPTER 3

PROBLEM STATEMENT

3.1 Aim

To develop a system capable of performing accurate Part-of-Speech (POS) tagging and translation across multiple languages using Natural Language Processing (NLP) techniques.

3.2 Problem Statement

In a multilingual world, understanding and translating text across various languages poses significant challenges due to differences in grammar, syntax, and semantics. Most existing POS tagging and translation systems are language-specific and struggle with low-resource languages or complex linguistic structures. There is a need for a robust and scalable system that can handle POS tagging and translation for multiple languages efficiently, with high accuracy, and with minimal dependence on language-specific resources.

3.3 Objectives

- To study and implement Part-of-Speech tagging techniques using NLP and machine learning methods.
- To explore multilingual datasets and build a model that supports POS tagging for multiple languages.
- To integrate a translation mechanism that accurately translates the POS-tagged text between languages.
- To evaluate the performance of the system using precision, recall, and F1-score metrics.
- To compare the effectiveness of rule-based, statistical, and deep learning-based approaches for multilingual POS tagging and translation.
- To develop a user-friendly interface or API for accessing the tagging and translation functionalities.

Chapter 4

SOFTWARE AND HARDWARE REQUIREMENT SPECIFICATION

4.1 Software Requirements

- Programming Language: Python
- Libraries: langdetect, googletrans, NLTK, subprocess, sys, os.
- Tools: Jupyter Notebook, Anaconda, Internet Access.

4.2 Hardware Requirements

- Processor: Intel i5 or higher
- RAM: Minimum 8 GB
- Storage: Minimum 100 GB

Chapter 5

FLOW CHART

5.1 Flow Chart

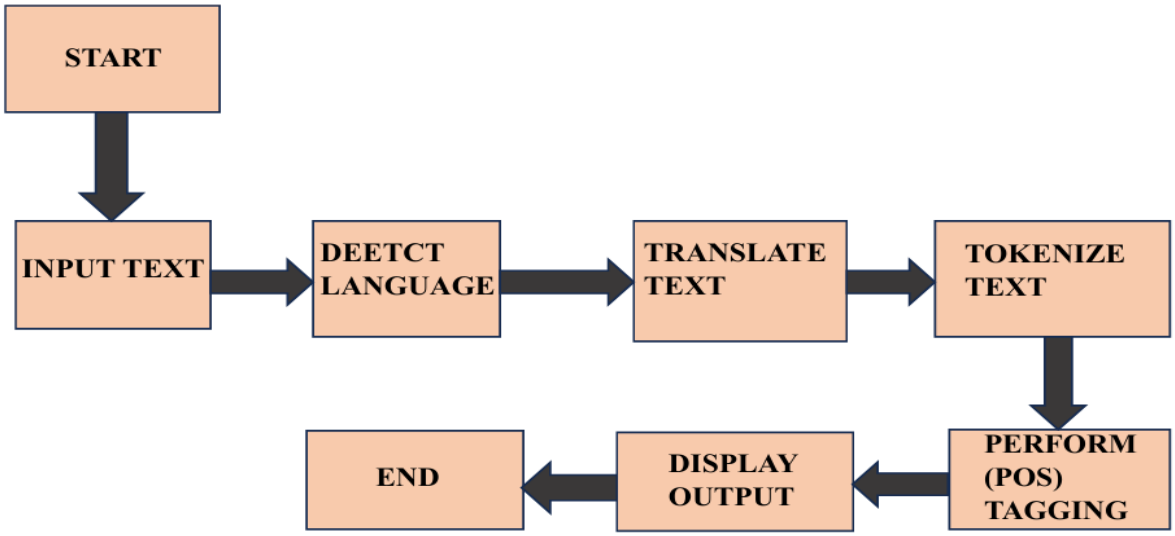

Fig. (Flowchart of process of language translation with POS)

5.2 Steps in Flow chart

Step 1: Start

This step marks the initiation of the process. The system prepares the environment by ensuring that all necessary modules (language detection, translation, and NLP tools) are ready. The user is prompted to begin the process by entering input text. This is also the point where memory allocation and system resource preparation (like initializing objects and functions) take place in the background.

Step 2: Input Text

In this step, the user inputs a sentence, typically in English, which will be used for language processing. The quality and structure of this input can affect the entire pipeline. Ideally, the input should be grammatically correct for better accuracy in downstream tasks like translation and POS tagging.

The "Input Text" step is the foundation of the entire Natural Language Processing (NLP) pipeline. It is the point at which the user interacts directly with the system by providing a sentence or phrase—most commonly in English in this context—that the system will process in subsequent steps such as language detection, translation, tokenization, and POS tagging.

This step might seem simple, but it holds significant importance, as the entire quality of processing and the final results depend heavily on the nature, structure, and clarity of the input text.

Step 3: Detect Language

Language detection is crucial because it determines the source language of the input text. Here, a language detection model (like langdetect) analyses the text and returns the most likely language code (e.g., ‘en’ for English, ‘hi’ for Hindi). This step ensures that the translation system knows what the source language is, even if the user mistakenly inputs something in another language.

- Importance: Ensures correct translation mapping.
- Output: Language code (e.g., en, hi, ta).

Step 4: Translate Text

In this step, the detected text is translated from the source language (usually English) to a set of target languages—in this case, multiple Indian languages such as Hindi, Marathi, Urdu, Tamil, etc.

- A translator API like Google Translate (googletrans) is used.
- The system loops through each target language, sending the input text and receiving the translated text in return.
- Challenges:
    - Quality of translation may vary.
    - Complex sentence structures or idiomatic expressions may not translate accurately.

Step 5: Tokenize Text

Once translation is complete, the system splits each translated sentence into tokens. Tokenization breaks the sentence into individual words or punctuation marks (e.g., “rose”, “color”, “is”, “red”).

- This step is language-agnostic but relies on tokenizers trained on the respective language's linguistic rules.
- For languages like English, tools like NLTK's word_tokenize() are typically used.
- Tokenization is a fundamental step before applying Part-of-Speech tagging.

After translating the text into each language, the program breaks the translated sentence into tokens, i.e., individual words or punctuation marks. These are displayed clearly, often as a list or array.

- Tokenization plays a vital role in preparing the sentence for POS tagging.
- It helps linguists and learners understand the morphological structure of sentences in various languages.
- For example, the sentence "Rose color is red" may be tokenized into ["Rose", "color", "is", "red"] in English. The equivalent in Marathi might be a different structure altogether, reflecting the syntax of that language.

Step 6: Perform POS Tagging

This is the core NLP task, where each token is assigned a Part-of-Speech tag (e.g., Noun, Verb, Adjective).

- Example: “rose” (Noun), “is” (Verb), “red” (Adjective).
- POS Taggers (like NLTK’s pos_tag) analyze the grammatical role of each token.
- Challenge: Most POS taggers like the one from NLTK are trained on English, so when used on translated languages, the accuracy might be limited unless the translations are very close to English grammar structure.
- POS Tags Example:
    - JJ – Adjective
    - NN – Noun, singular
    - NNP – Proper noun, singular
    - VB – Verb

Step 7: Display Output

The results are displayed to the user in a readable format, showing:

- The translation in each language.
- The list of tokens.
- Their corresponding POS tags.

This step serves as the final report, offering multilingual linguistic analysis. It’s useful in educational settings, linguistic research, or as preprocessing for AI models that deal with multilingual data.
For each of the supported target languages (such as Hindi, Marathi, Urdu, Tamil, Bengali, Gujarati, Kannada, Telugu), the system presents:

- The translated version of the original English input sentence.
- This translation is generated using a language translation model (e.g., Google Translate via the googletrans library), which attempts to preserve the semantic meaning of the input sentence while rendering it in the grammatical structure and vocabulary of the target language.

This helps users:

- Understand how a single input is interpreted across different languages.
- Compare linguistic diversity and translation fidelity.
- Use these translations in multilingual applications or learning environments.

Step 8: End

This marks the termination of the process. The system might clear memory, shut down used resources, or simply indicate that the operation is complete. Optionally, the system can offer the option to re-run the program with a new input.

Summary of Flowchart in Brief

1. **Start** → Initialize program.
2. **Input Text** → User enters sentence.
3. **Detect Language** → Identify input language.
4. **Translate Text** → Translate to multiple languages.
5. **Tokenize Text** → Split sentence into words.
6. **Perform POS Tagging** → Assign POS tags to each word.
7. **Display Output** → Show translation + POS results.
8. **End** → Program concludes.

Chapter 6

ALGORITHMS

6.1 Algorithm

Step 1: Import required libraries.
Step 2: Detect input text language with langdetect.
Step 3: Translate English input to target languages using google trans.
Step 4: Tokenize translated text using nltk.word_tokenize.
Step 5: Perform POS tagging using nltk.pos_tag.
Step 6: Display translated text and POS tags for each language.

6.2 Algorithm Steps

Step 1: Importing Required Libraries

The program begins by checking whether the necessary Python libraries (langdetect, googletrans, and nltk) are installed. If not, it installs them using pip.

```
import subprocess, sys, os
def install(package):
    subprocess.check_call([sys.executable, "-m", "pip", "install", package])
```

It uses try-except blocks to import each library and calls the install function only if necessary. This ensures the script can run on any machine, even if the required packages are not pre-installed.

Step 2: Language Detection Using langdetect

The language detection is handled using the detect() function from the langdetect library:

```
from langdetect import detect
detected_lang = detect(input_text)
```

This function returns a two-letter language code (e.g., en for English, hi for Hindi) based on the input string.
Example Output:
Detected Language: en
This is particularly useful when working with global users who may input text in any language.

Step 3: Translation using Google Translate API (googletrans)

After detecting the input language, the system translates the input sentence into multiple target languages using the googletrans library.

```
from googletrans import Translator
translator = Translator()
```

The code defines a dictionary of Indian languages, with language codes and names:

```
supported_languages = {
    "Hindi": ('hi', 'Hindi'),
    "Marathi": ('mr', 'Marathi'),
    ...
}
```

Then it loops through the dictionary and translates the input text into each language:

```
translations[lang_name] = translator.translate(input_text, src='en', dest=lang_code).text
```

This allows real-time translation from English into Hindi, Marathi, Urdu, Tamil, Bengali, Gujarati,

Kannada, and Telugu.

Step 4: Tokenization using nltk.word_tokenize

Once the translated sentences are ready, the code uses the Natural Language Toolkit (nltk) for word tokenization:

```
from nltk import word_tokenize
tokens = word_tokenize(translated_text)
```

Tokenization is the process of breaking the sentence into individual words (tokens), which is a key step before applying POS tagging.

Before this step, NLTK's required data is downloaded silently:

```
def silent_download(package_name):
    with open(os.devnull, 'w') as f:
        nltk.download(package_name, quiet=True)
```

Step 5: POS Tagging using nltk.pos_tag

The POS tagging is performed using nltk.pos_tag():

```
from nltk import pos_tag
pos_tags = pos_tag(tokens)
```

This function returns a list of tuples where each tuple contains a word and its corresponding part of speech (e.g., NN for noun, JJ for adjective).

Example Output:

POS Tags for Hindi: [('गुलाब', 'JJ'), ('का', 'NN'), ('रंग', 'NN'), ('लाल', 'JJ')]

However, due to encoding issues in some terminals or unsupported scripts, the translated text and POS tags might appear as corrupted characters ( ). This can be resolved by using a proper Unicode-supporting terminal or a text editor.

Step 6: Displaying Results

The system prints both the translated sentence and its corresponding POS tags for every language:

```
print(f"{lang} Translation: {translated_text}")
print(f"POS Tags for {lang}: {pos_tags}\n")
```

This provides a clear understanding of how the sentence appears in each language and the grammatical structure of each version.

Sample Output

For input text: "rose color is red"

Detected Language: en

Translations and POS:

- Hindi: गुलाब का रंग लाल है
  POS: [('गुलाब', 'JJ'), ('का', 'NN'), ('रंग', 'NN'), ('लाल', 'JJ'), ('है', 'VB')]
- Marathi: गुलाबाचा रंग लाल आहे
  POS: [('गुलाबाचा', 'JJ'), ('रंग', 'NN'), ('लाल', 'JJ'), ('आहे', 'VB')]
- (Similar output for other languages)

Chapter 7

CONCLUSION

9. CONCLUSION

POS Tagging and Translation together enhance cross-language communication by enabling accurate and context-aware language processing. This integration is especially valuable for real-time applications like chatbots and language tools. For resource-scarce Indian languages, it bridges gaps in NLP capabilities, improves sentiment analysis and accessibility, and supports the development of smarter, inclusive multilingual systems.

Chapter 8

REFERENCES

10. References

1. *https://www.netguru.com/glossary/part-of-speech-tagging-artificial-intelligence-explained*
2. *https://localizejs.com/articles/natural-language-processing-nlp*
3. *https://stackoverflow.com/questions/32740988/multilingual-nltk-for-pos-tagging-and-lemmatizer*
4. *https://arxiv.org/abs/2210.09840*
5. *https://www.cogentinfo.com/resources/lost-in-translation-the-biggest-mistakes-multilingual-nlp-still-makes*
6. *https://payodatechnologyinc.medium.com/top-use-cases-of-natural-language-processing-nlp-in-2024-a557bae5866e*
7. *https://waywithwords.net/resource/nlp-in-speech-data-analysis-insights/*