AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department Of Computer Engineering

An NLP PROJECT REPORT

ON

POS TAGGERS FOR INDIAN LANGUAGES

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERINGAISSMS IOIT

BE Computer Engineering

SUBMITTED BY

STUDENT NAME	ERP No:
Onasvee Banarse	**09**
Kaustubh Kabra	**37**
Akash Mete	**50**
Harsh Shah	**67**

2022 -2023

AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report.
"POS TAGGERS FOR INDIAN LANGUAGES"

Submitted by

STUDENT NAME	ERP No:
Onasvee Banarse	09
Kaustubh Kabra	37
Akash Mete	50
Harsh Shah	67

is a bonafide student at this institute and the work has been carried out by them under the supervision of **Prof. P.D.Bormane** and it is approved for the partial fulfillment of the Department of Computer Engineering AISSMS IOIT.

(**Prof. P.D.Bormane**) (**Dr. S.N.Zaware**)

Mini-Project Guide Head of Computer Department

Place: Pune Date:

Abstract

The Mini project aims to conduct a comparative study of POS taggers for Indian languages, focusing on Hindi, Bengali, Tamil, and Telugu. The project will evaluate the accuracy, coverage, and applicability of different taggers available for these languages. A diverse corpus of text data will be collected, pre-processed, and tagged using existing POS taggers such as IndicNLP, NLTK, TreeTagger, Stanford POS Tagger, and ILMT. Performance metrics such as accuracy, precision, recall, and F1-score will be calculated to assess the quality of the taggers.

The project will analyze and compare the performance of the taggers, identify common errors, and provide recommendations for selecting the most suitable POS tagger for each Indian language. The outcomes will facilitate improved understanding and usage of POS taggers in Indian language processing.

Contents

Introduction

This Mini project focuses on conducting a comparative study of POS taggers for Indian languages, specifically Hindi, Bengali, Tamil, and Telugu. POS tagging, an essential task in Natural Language Processing (NLP), involves assigning grammatical labels to words in a sentence. While POS taggers have been extensively developed for English, their availability and performance for Indian languages are relatively limited. Indian languages pose unique linguistic challenges due to rich morphology, complex sentence structures, and inflections. This project aims to evaluate the accuracy, coverage, and applicability of existing POS taggers for Indian languages, providing insights and recommendations for selecting the most suitable tagger for each language.

The project methodology involves collecting diverse text data from sources like news articles, social media, and literature in the target languages. This data will be preprocessed and tagged using different POS taggers, including IndicNLP, NLTK, TreeTagger, Stanford POS Tagger, and ILMT. Performance metrics such as accuracy, precision, recall, and F1-score will be calculated.

In summary, this mini project seeks to evaluate and compare the performance of POS taggers for Indian languages, focusing on Hindi, Bengali, Tamil, and Telugu. The project aims to provide valuable insights and recommendations for selecting the most suitable tagger for each language, addressing the linguistic challenges specific to Indian languages and enhancing language processing capabilities in these diverse linguistic contexts.

Software Requirement Specification

Software Used:

- VScode –

  Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

- Python (version 3 or above)-

  Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. The sentiment analysis is performed using python language and packages.

- Jupyter Notebook or Google Colab –

  Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages".

- NLTK (Natural Language Toolkit) –

  NLTK (Natural Language Toolkit) is a widely used Python library for natural language processing. It provides a wide range of tools, resources, and algorithms for text processing and analysis. NLTK offers support for various NLP tasks, including tokenization, stemming, lemmatization, POS tagging, parsing, and more.

Hardware Requirement

The detailed hardware requirements for the project are:

Item	Description
System	HP OMEN 15 series
Processor	AMD Ryzen 5 4600H
RAM	8 GB
System Type	64-bit operating system, x64-based processor
SSD	256 GB Solid State Drive
HDD	1 TB Hard Disk Drive
Graphics	NVIDIA 4 GB Graphic Card
Operating System	Windows 10 Operating System

Theory

Part of Speech (POS) Tagging is the first step in the development of any NLP Application. It is a task which assigns POS labels to words supplied in the text. This is the reason why researchers consider this as a sequence labeling task where words are considered as sequences which needs to be labeled. Each word's tag is identified within a context using the previous word/tag combination. POS tagging is used in various applications like parsing where word and their tags are transformed into chunks which can be combined to generate the complete parse of a text.

Taggers are used in Machine Translation (MT) while developing a transfer based MT Engine. Here, we require the text in the source language to be POS tagged and then parsed which can then be transferred to the target side using transfer grammar. Taggers can also be used in Name Entity Recognition (NER) where a word tagged as a noun (either proper or common noun) is further classified as a name of a person, organization, location, time, date etc.

Tagging of text is a complex task as many times we get words which have different tag categories as they are used in different context. This phenomenon is termed as lexical ambiguity. For example, let us consider text in Table 1. The same word 'सोना' given a different label in the two sentences. In the first case it is termed as a common noun as it is referring to an object (Gold Ornament). In the second case it is termed as a verb as it is referring to an experience (feelings) of the speaker. This problem can be resolved by looking at the word/tag combinations of the surrounding words with respect to the ambiguous word (the word which has multiple tags).

Over the years, a lot of research has been done on POS tagging. Broadly, all the efforts can be categorized in three directions. They are: rule based approach where a human annotator is required to develop rules for tagging words or statistical approach where we use mathematical formulations and tag words or hybrid approach which is partially rule based and partially statistical. In the context of European languages POS taggers are generally developed using machine learning approach, but in the Indian context, we still do not have

a clear good approach. In this paper we discuss the development of a POS tagger for Hindi using Hidden Markov Model (HMM).

सोने	के	आभूषण	महंगे	हो	गए	है
NN	PSP	NN	JJ	VM	VAUX	VAUX
उसका	दिल	**सोने**	का	है		
PRP	NN	VM	PSP	VM		

Table 1. Example of Lexical Ambiguity

IMPLEMENTATION & IMPORTANT MODULES

In this project we are doing POS tagging for Hindi sentences and for that we have used **Python**.

For developing a HMM based tagger we were first required to annotate a corpus based on a tagset.

Modules

Downloading dataset

So, using our source code we first download a Hindi dataset which has numerous sentence in Hindi.

Preprocessing the downloaded dataset

Our next step is to preprocess the corpus dataset which we have downloaded so as to implement operations on it. This is done by selecting every individual sentence from the dataset for which we want POS tagging.

Stripping words in sentence

Following the previous step, we strip the words from the sentence so that separate operations can be performed. For example, we take a sentence and strip it, we then perform operation on each constituting word to tag it with the most accurate POS tags.

Training POS tagger

This way we achieve the module of training the POS tagger with the results thus obtained for each and every data in the corpus. Now the POS tagger is ready to tag any Hindi sentence.

Tagging new line

At the end when the POS tagger is trained, it can then be used for tagging new Hindi lines, according to the user's choice.

Implementing Concepts

A POS tagger based on HMM assigns the best tag to a word by calculating the forward and backward probabilities of tags along with the sequence provided as an input. The following equation explains this phenomenon.

$$P(t_i|w_i) = P(t_i|t_{i-1}).P(t_{i+1}|t_i).P(w_i|t_i) \qquad (1)$$

Here $P(t_i|t_{i-1})$ is the probability of a current tag given the previous tag and $P(t_{i+1}|t_i)$ is the probability of the future tag given the current tag. This captures the transition between the tags.

These probabilities are computed using equation 2.

$$P(t_i|t_{i-1}) = \frac{freq\ (t_{i-1}, t_i)}{freq(t_{i-1})} \qquad (2)$$

Each tag transition probability is computed by calculating the frequency count of two tags seen together in the corpus divided by the frequency count of the previous tag seen independently in the corpus. This is done because we know that it is more likely for some tags to precede the other tags. For example, an adjective (JJ) will be followed by a common noun (NN) and not by a postposition (PSP) or a pronoun (PRP). Figure 1 shows this example.

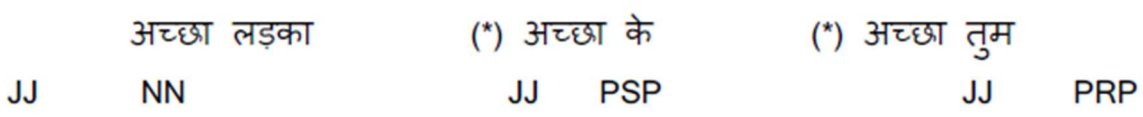

Figure 1. Tag transition probabilities

POS Tags for Hindi sentences

S.No.	Tag	Description (Tag Used for)	Example
1.	NN	Common Nouns	लड़का, लड़के, किताब, पुस्तक
2.	NST	Nourn Denotating Spatial and Temporal Expressions	ऊपर, पहले, बहार, आगे
3.	NNP	Proper Nourns (name of person)	मोहन, राम, सुरेश
4.	PRP	Pronoun	वह, वो, उसे, तुम
5.	DEM	Demonstrative	वह, वो, उस
6.	VM	Verb Main (Finite or Non-Finite)	खाता, सोता, रोता, खाते, सोते, रोते
7.	VAUX	Verb Auxilary (Any verb, present besides main verb shall be marked as auxillary verb)	है, हुए, कर
8.	JJ	Adjective (Modifier of Noun)	सांस्कृतिक, पुरानी, दुपहिया
9.	RB	Adverb (Modifier of Verb)	जल्दी, धीरे, धीमे
10.	PSP	Postposition	में, को, ने
11.	RP	Particles	भी, तो, ही
12.	QF	Quantifiers	बहुत, थोडा, कम
13.	QC	Cardinals	एक, दो, तीन
14.	CC	Conjuncts (Coordinating and Subordinating)	और, की
15.	WQ	Question Words	क्यों, क्या, कहा
16.	QO	Ordinals	पहला, दूसरा, तीसरा
17.	INTF	Intensifier	बहुत, थोडा, कम
18.	INJ	Interjection	अरे, हाय
19.	NEG	Negative	नहीं, ना
20.	SYM	Symbol	?, ;: !
21.	XC	Compounds	केंद्र/XC सरकार/NN रंग/XC बिरंगे/JJ
22.	RDP	Reduplications	धीरे/RB धीरे/RDP

Code

Code:

```
import nltk
from nltk.corpus import indian
from nltk.tag import tnt
import string


nltk.download('punkt')
nltk.download()

tagged_set = 'hindi.pos'
word_set = indian.sents(tagged_set)
count = 0
for sen in word_set:
    count = count + 1
    sen = "".join([" "+i if not i.startswith("'") and i not in string.punctuation else i for i in
sen]).strip()
    print (count, sen)
print ('Total sentences in the tagged file are',count)

train_perc = .9
train_rows = int(train_perc*count)
test_rows = train_rows + 1

print ('Sentences to be trained',train_rows, 'Sentences to be tested against',test_rows)

data = indian.tagged_sents(tagged_set)
train_data = data[:train_rows]
test_data = data[test_rows:]

pos_tagger = tnt.TnT()
pos_tagger.train(train_data)
pos_tagger.evaluate(test_data)

sentence_to_be_tagged = "३९ गेंदों में दो चौकों और एक छक्के की मदद से ३४ रन बनाने वाले परोरे
अंत तक आउट नहीं हुए ।"

tokenized = nltk.word_tokenize(sentence_to_be_tagged)
print(pos_tagger.tag(tokenized))

print(data)
```

Output

```
In [8]: sentence_to_be_tagged = "३९ गेंदों में दो चौकों और एक छक्के की मदद से ३४ रन बनाने वाले परोरे अंत तक आउट नहीं हुए ।"

tokenized = nltk.word_tokenize(sentence_to_be_tagged)


print(pos_tagger.tag(tokenized))
```

```
[('३९', 'QFNUM'), ('गेंदों', 'NN'), ('में', 'PREP'), ('दो', 'QFNUM'), ('चौकों', 'QFNUM'), ('और', 'CC'), ('एक', 'QFNUM'), ('छक्के', 'QF
NUM'), ('की', 'PREP'), ('मदद', 'NN'), ('से', 'PREP'), ('३४', 'QFNUM'), ('रन', 'NN'), ('बनाने', 'VNN'), ('वाले', 'PREP'), ('परोरे',
'NNP'), ('अंत', 'Unk'), ('तक', 'PREP'), ('आउट', 'JVB'), ('नहीं', 'NEG'), ('हुए', 'VAUX'), ('।', 'PUNC')]
```

```
In [27]: df=pd.DataFrame(data)
         df.iloc[:, 0:11]
```

Out[27]:

	0	1	2	3	4	5	6	7	8	9	10
0	(पूर्ण, JJ)	(प्रतिबंध, NN)	(हटाओ, VFM)	(:, SYM)	(इराक, NNP)	None	None	None	None	None	None
1	(संयुक्त, NNC)	(राष्ट्र, NN)	(।, SYM)	None	None	None	None	None	None	None	None
2	(इराक, NNP)	(के, PREP)	(विदेश, NNC)	(मंत्री, NN)	(ने, PREP)	(अमरीका, NNP)	(के, PREP)	(उस, PRP)	(प्रस्ताव, NN)	(का, PREP)	(मजाक, NVB)
3	(विदेश, NNC)	(मंत्री, NN)	(का, PREP)	(कहना, VFM)	(है, VAUX)	(कि, CC)	(चूंकि, CC)	(बगदाद, NNP)	(संयुक्त, NNC)	(राष्ट्र, NN)	(की, PREP)
4	(लिहाजा, CC)	(प्रतिबंधों, NN)	(को, PREP)	(पूर्ण, JJ)	(रूप, NN)	(से, PREP)	(उठा, VFM)	(दिया, VAUX)	(जाना, VAUX)	(चाहिए, VAUX)	(।, PUNC)
...	...	...	...	...	...	...	...	...	...	...	...
535	(अनवर, NNP)	(और, CC)	(नजीर, NNP)	(ने, PREP)	(पहले, JJ)	(विकेट, NN)	(के, PREP)	(लिए, PREP)	(४५, QFNUM)	(और, CC)	(फिर, RB)
536	(पाक, NNP)	(के, PREP)	(पुछल्ले, JJ)	(बल्लेबाजों, NN)	(में, PREP)	(अजहर, NNPC)	(महमूद, NNP)	(ने, PREP)	(१७, QFNUM)	(रन, NN)	(बनाये, VFM)
537	(कप्तान, NN)	(मोइन, NNPC)	(खान, NNP)	(ने, PREP)	(१५, QFNUM)	(रनों, NN)	(का, PREP)	(योगदान, NN)	(दिया, VFM)	(।, PUNC)	None
538	(न्यू, NNPC)	(जीलैंड, NNP)	(की, PREP)	(तरफ, NN)	(से, PREP)	(टफी, NNP)	(और, CC)	(फ्रैंक्लीन, NNP)	(ने, PREP)	(तीन-तीन, QFNUM)	(और, CC)
539	(अनवर, NNP)	(को, PREP)	(विंसेट, NNP)	(ने, PREP)	(रन, NNC)	(आउट, NN)	(किया, VFM)	(।, PUNC)	None	None	None

540 rows × 11 columns

Conclusion

In conclusion, the Mini project conducted a comparative study of POS taggers for Indian languages, specifically focusing on Hindi, Bengali, Tamil, and Telugu. The project evaluated the accuracy, coverage, and applicability of different existing taggers and provided recommendations for selecting the most suitable tagger for each language.

Through the analysis of a diverse corpus of text data and the utilization of various POS taggers such as IndicNLP, NLTK, TreeTagger, Stanford POS Tagger, and ILMT, the project assessed the performance of these taggers. By comparing the tagged output with manually annotated gold standard data, performance metrics such as accuracy, precision, recall, and F1-score were calculated.

The findings of the project highlighted the strengths and weaknesses of each tagger in handling the linguistic complexities of Indian languages. It also identified common errors and challenges faced by the taggers, providing insights for potential improvements. The project's recommendations aimed to guide researchers and practitioners in selecting the most suitable POS tagger for each Indian language, facilitating more accurate and efficient analysis of text data in these languages.

Overall, this project contributes to the advancement of NLP research in Indian languages by evaluating and comparing the performance of POS taggers. It provides insights and recommendations that can enhance language processing capabilities in Indian languages, aiding in a more effective understanding and analysis of text data.

References

Books:

1. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition" by Daniel Jurafsky and James H. Martin.
2. "Foundations of Statistical Natural Language Processing" by Christopher D. Manning and Hinrich Schütze.
3. "Natural Language Processing with Python" by Steven Bird, Ewan Klein, and Edward Loper.
4. "Introduction to Information Retrieval" by Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze.
5. "Handbook of Natural Language Processing" edited by Nitin Indurkhya and Fred J. Damerau.

Websites:

1. Natural Language Toolkit (NLTK): https://www.nltk.org/
2. Stanford NLP: https://nlp.stanford.edu/
3. Kaggle: https://www.kaggle.com/
4. Towards Data Science: https://towardsdatascience.com/
5. ACL Anthology: https://www.aclweb.org/anthology/