Indian Institute of Science Education and Research Bhopal

(Dept. of Data Science And Engineering)

A Project Report on

"HMM-Based POS-Tagger for Hindi Language"

Submitted in fulfillment of the course project for the course

Computational Linguistics(DSE 308)

by

Deep Pooja-17074

(Dept. of EECS, IISER Bhopal)

Under the guidance
Of

Dr. Rajakrishnan P. Rajkumar

Assistant Professor
Department of HSS, IISER Bhopal
Bhopal-462066

TABLE OF CONTENTS

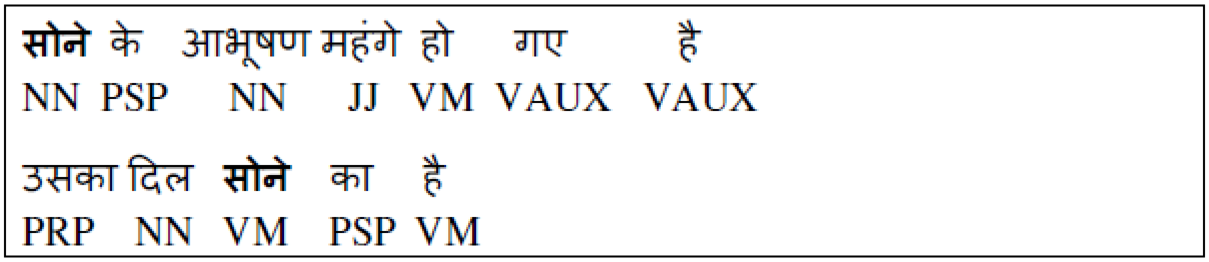
INTRODUCTION

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 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 in Hindi language (This example I have taken from a reliable source on the internet), let us consider text in Table 1. The same word 'सोना' given a different label in the two sentences. In the first sentence it is termed as a common noun as it is referring to an object (Gold Ornament) and in the second sentence 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 (Context words) with respect to the ambiguous word (like the word 'सोना' here).

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 report I will discuss the development of a POS tagger for Hindi language 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 I have worked on POS tagging for Hindi language and for that I have chosen **Python programming language**.
For developing a HMM based tagger it was first required to annotate a corpus based on a tag set.

Modules

Downloading dataset

I have used Indian dataset with hindi.pos tag set from NLTK, which can easily downloaded/imported from NTTK corpus.

Preprocessing the downloaded dataset

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

Stripping words in sentence

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

Training POS tagger

Now comes the training module, first the preprocessed data should be split into train set and test set. I have used in-built HMM implemented function from NLTK which is TnT.

TnT uses a second order Markov model to produce tags for a sequence of input, specifically:

argmax [Proj(P(t_i|t_i-1,t_i-2)P(w_i|t_i))] P(t_T+1 | t_T)

IE: the maximum projection of a set of probabilities

The set of possible tags for a given word is derived from the training data. It is the set of all tags that exact word has been assigned.

To speed up and get more precision, we can use log addition
to instead multiplication, specifically:

    argmax [Sigma(log(P(t_i|t_i-1,t_i-2))+log(P(w_i|t_i)))] +
        log(P(t_T+1|t_T))

The probability of a tag for a given word is the linear interpolation of 3 Markov models; a zero-order, first-order, and a second order model.

    P(t_i| t_i-1, t_i-2) = l1**P(t_i) + l2**P(t_i| t_i-1) +
                    l3*P(t_i| t_i-1, t_i-2)

Now the 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 sentences.

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 concept.

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

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.

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

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

अच्छा	लड़का	(*)	अच्छा	के	(*)	अच्छा	तुम
JJ	NN		JJ	PSP		JJ	PRP

Figure 1. Tag transition probabilities

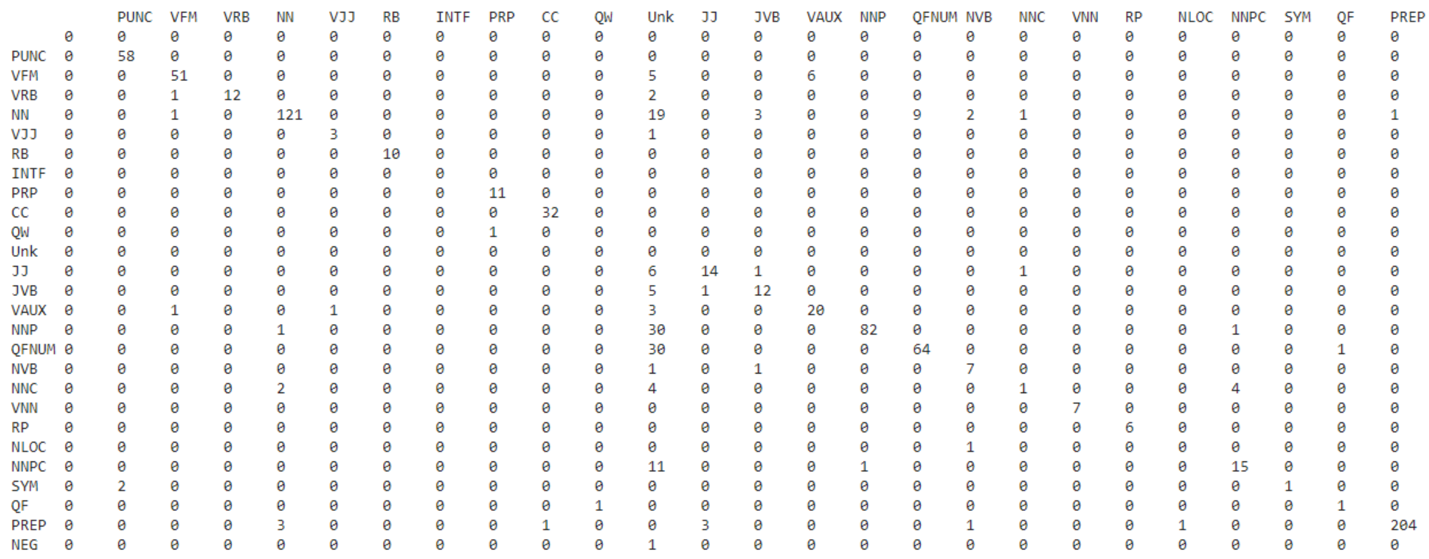
POS Tags for Hindi sentences

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

Computing Confusion Matrix

It is a measure of where the classifier, here our tagger is doing wrong. To compute confusion matrix we have to count how many times labels are predicted in comparison to an expected gold standard. Then use that information to give meaningful metrics. A confusion matrix has counts of how many times labels are predicted by a tagger against a gold standard.



		PUNC	VFM	VRB	NN	VJJ	RB	INTF	PRP	CC	QW	Unk	JJ	JVB	VAUX	NNP	QFNUM	NVB	NNC	VNN	RP	NLOC	NNPC	SYM	QF	PREP
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PUNC	0	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
VFM	0	0	51	0	0	0	0	0	0	0	0	5	0	0	6	0	0	0	0	0	0	0	0	0	0	0
VRB	0	0	1	12	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NN	0	0	1	0	121	0	0	0	0	0	0	19	0	3	0	0	9	2	1	0	0	0	0	0	0	1
VJJ	0	0	0	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RB	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INTF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PRP	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CC	0	0	0	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
QW	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Unk	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
JJ	0	0	0	0	0	0	0	0	0	0	0	6	14	1	0	0	0	0	1	0	0	0	0	0	0	0
JVB	0	0	0	0	0	0	0	0	0	0	0	5	1	12	0	0	0	0	0	0	0	0	0	0	0	0
VAUX	0	0	1	0	0	1	0	0	0	0	0	3	0	0	20	0	0	0	0	0	0	0	0	0	0	0
NNP	0	0	0	0	1	0	0	0	0	0	0	30	0	0	0	82	0	0	0	0	0	0	1	0	0	0
QFNUM	0	0	0	0	0	0	0	0	0	0	0	30	0	0	0	0	64	0	0	0	0	0	0	0	1	0
NVB	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	7	0	0	0	0	0	0	0	0
NNC	0	0	0	0	2	0	0	0	0	0	0	4	0	0	0	0	0	0	1	0	0	0	4	0	0	0
VNN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0
RP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0
NLOC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
NNPC	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	1	0	0	0	0	0	0	15	0	0	0
SYM	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
QF	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
PREP	0	0	0	0	3	0	0	0	0	1	0	0	3	0	0	0	0	1	0	0	0	1	0	0	0	204
NEG	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Now we can also compute F1-score, precision and Recall for the tagger we just built for Hindi language.

True Positive : TPTP

Predicted label that matches the expected gold label.

These are the diagonal counts of the *confusion matrix*.

False Negative : FNFN
All expected gold labels that don't match the predicted label. All the incorrect golds for a label.

Sometimes called a Type II Error.

This is the sum of the gold label's row without the predicted counts in the *confusion matrix*.

False Positive : FPFP
All prediction labels that don't match the expected gold label. All the incorrect predictions for a label.

Sometimes called a Type I Error.

This is the sum of the predicted label's column without the gold counts in the *confusion matrix*.

True negative : TNTN
All remaining labels that don't match the predicted or gold labels.

This is the sum of the counts that aren't in the row or column of the label in question of the *confusion matrix*.

Accuracy

Accuracy is how many predictions were right.

$$\text{Acc} = \frac{TP+TN}{TP+TN+FP+FN}$$

This is the metric that is usually used first and easily calculated, it's how many correct predictions were made out of the total.

Precision

Precision is how many predictions are relevant.

$$\text{P} = \frac{TP}{TP+FP}$$

A good metric for determining how many correct predictions were made out of the right and wrong ones.

Recall

Recall is how many relevant predictions were made.

$$R = \frac{TP}{TP+FN}$$

A good metric for determining how many correct predictions were made out of the right and incorrectly marked right ones.

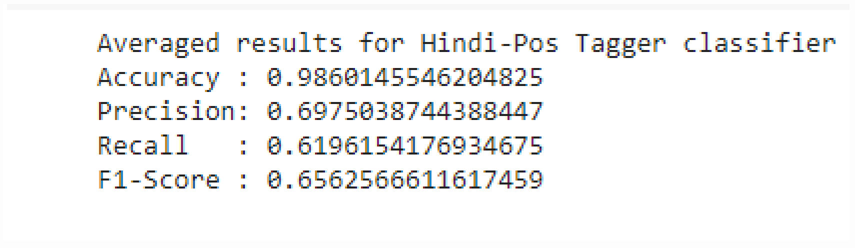
F1-Score

F1-Score is a mix of precision and recall. A harmonic mean of the two (where β=1β=1).

$$F1 = \frac{2PR}{P+R}$$

A nice metric for capturing how both metrics together. You can really only maximize one of the two due to the precision-recall curve observation, therefore the F1-Score is a nice value that can be used for optimization or both simultaneously

Here are the results from our tagger:



```
Averaged results for Hindi-Pos Tagger classifier
Accuracy : 0.9860145546204825
Precision: 0.6975038744388447
Recall   : 0.6196154176934675
F1-Score : 0.6562566611617459
```

Conclusion:

Since we have trained our tagger on specific dataset and hence it is not generalized to tag sentences which have vocabulary out of training set, the tagger simply tag it to unknown words instead of giving any POS tag.

Confusion matrix is also sparse. Most of the wrong tagging is due to unknown noun words, also tagger is confused between proper noun and common noun.

We can further improve the tagger using more data. This work can be extended to Name entity recognition for Hindi language or even parsing the language or machine translation.