IJCRT.ORG ISSN : 2320-2882
INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

# Gujarati Language POS Tagging Using Hidden Markov Model (HMM)

Dr. Dikshan Shah
Vanita Vishram Women's University, Surat, Gujarat

***Abstract***
Part of Speech (POS) tagging refers to the process of classifying each morpheme, including punctuation marks, in a particular text document according to the context. An important first step in natural language processing is the parts of speech tagging (POS) process (NLP). To tag each word in the corpus with its appropriate parts of speech is its goal. Noun, pronoun, verb, adjective, and adverb are the fundamental POS tags, among others. POS tags are required for speech recognition and analysis, machine translation, lexical analysis such as word sense disambiguation, named entity recognition, information retrieval, and this system also assisted opinion mining by revealing the sentiments of a given text. POS taggers are also lacking in many Indian languages because basic resources like corpora and morphological analyzers are still being researched and developed. The following section of this research proposes a probabilistic Hidden Markov Model-based POS tagger for Gujarati. In order to reduce ambiguity and misclassification rates, the hidden Markov model predicts the hidden sequence based on the highest observation likelihood. A variety of POS tags at the word level make up the model that was tested using sample input text data.

**Index Terms - Indian Languages, POS tagging, Hidden Markov Model, Probabilistic approach**

## I. INTRODUCTION

More than 19,500 languages and dialects are spoken as first languages in the multicultural country of India.[3] According to the Registrar General and Census Commissioner, 96.71% of Indians spoke one of the 22 languages that were slated for official use in 2018. More than 50 million people speak Gujarati, a language that is indigenous to the Gujarat state and one of the 22 languages recognised by the Constitution.[2][3]

In the technology age, Indians have begun to use their regional tongues when utilising social media. By offering programmes in their native or regional languages, even well-known digital juggernauts like Microsoft, Apple, Google, and Amazon keep their consumers satisfied. Thus, the development of NLP (Natural Language Processing) is crucial for regional languages.

In order to produce a solution and develop methods for teaching computers to understand and speak natural languages, NLP research aims to gain a better understanding of how people use and perceive language. The roots of NLP can be found in languages, psychology, electrical and electronic engineering, computer and information sciences, robotics, and mathematics, among other disciplines.

A method for labelling each word in a text document with the appropriate part of speech is called part-of-speech (POS) tagging.[9] The amount of data that POS tags, word classes, morphological classes, and lexical tags generate about a word and its neighbours emphasises the importance of these features for language processing. POS tagging can be used for text-to-speech, information extraction, information retrieval, and corpus-based linguistic analysis.[7] This makes POS tagging the principal use of extended NLP in Indian languages. An exhaustively trained HMM-based POS tagging technique for Gujarati is presented in this study.

## II. CHALLENGES OF NER IN INDIAN LANGUAGES

For South and South East Asian languages, the NER issue is still unresolved. Precise named entity recognition algorithms are already available for European languages, particularly English.[3] A crucial demand in the digital age is NER in Indian languages. Despite the fact that Indian languages face many tough issues, naming named entities in Indian languages is more complicated than it is in non-Indian languages because of capitalization, a lack of resources, ambiguity, morphological richness, etc.[2] Due to the differences in each language's syntax and semantics, no existing NER approach can be used to Indian languages. Following are a few of the significant difficulties:

- **AGGLUTINATION**

The languages of India are agglutinative. In order to create a word, case markers are thus added to proper or common nouns. As an example: મા, થી, ની.With more case indicators being suffixed to nouns, it is harder for the machine to comprehend various NE patterns.

- **AMBIGUITY**

In Indian languages, there is a lot of ambiguity between a proper noun and a common noun. As an example: ગાંગા, કાવેરી. These are the names of a person as a proper noun and the names of a river as a common noun.

- **SPELL VARIATIONS**

A word can be represented in various ways while yet having only one meaning. The hardest problem to solve is how to spell the same thing differently. As an example: વાજપેયી, વાજપેય, વાજપેઈ.

- **NO CAPITALIZATION**

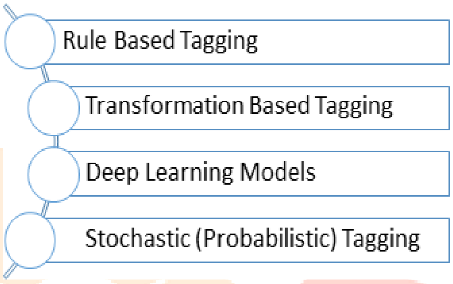
Indian languages have their own morphology and do not follow capitalization conventions.

- **COMPOUND NAMED ENTITIES**

A named entity is a form of a number of distinct entities. For example – 'યુનનયન બેંક ઓફ ઇનડિયા'. The name of the bank can be derived from the complete list of words.

## III. EXISTING TECHNIQUES

For POS tagging, a number of strategies can be utilized.




**[Fig .1 Existing POS Tagging Techniques]**

- **RULE-BASED POS TAGGING**

Rule-based POS tagging applies specially developed rules and annotates the words based on the context.[5] These laws are frequently referred to as "contextual laws." High levels of language competence are required to create effective regulations.

- **TRANSFORMATION BASED TAGGING**

This approach makes use of pre-established handwritten rules and mechanically generated training practices.

- **DEEP LEARNING MODELS**

For POS labelling, several Deep Learning models are also available. The Meta-BiLSTM model has a remarkable accuracy of about 97% when compared to other POS tagging models like LSTM, Vanilla RNN, GRU, Word Embedding, etc.

- **STOCHASTIC (PROBABILISTIC) TAGGING**

This method is additionally known as a statistical, frequency-based, or probability-based method. This method tags unannotated information by using the tag that appears the most in the training dataset for a specific word.[8] For a single word, many tag sequences were discovered, which is improper according to grammar rules. The described constraint can be solved by calculating the likelihood for several tag sequences and assigning the POS tag based on the sequence with the highest probability. A probabilistic model for POS tagging is the Hidden Markov model.

- **HIDDEN MARKOV MODEL**

Secret Markov Model, which allocates the combined probability to the label sequence and paired observations, is a probabilistic model. Then, in order to raise the overall likelihood of training sets, parameters are trained.

Formally, HMM is defined as follows:

$$\lambda = (A, B, \pi)$$

Where A - is the transition probability
B – Is the emission probability
π – Representing the start probability

$$A = a_{ij} = \frac{(\text{Number of transitions from state } s_i \text{ to } s_j)}{(\text{Number of transitions from state } s_i)}$$

$$B = b_j(k) = \frac{(\textbf{Number of times in state j and observing symbol k})}{(\textbf{expected number of times in state j})}$$

Sentences are where the word first appeared. To determine the mode estimates for various HMM parameters, the Baum Welch algorithm is used to determine the parameter with the highest likelihood. A sequence of observations or emissions is produced by using the Forward-Backward Algorithm to determine the successive marginal of all hidden state variables that are given.

- **VITERBI ALGORITHM**

In order to find the optimal likely tag sequence in the state space of the probable tag division based on the state transition probabilities, the Viterbi algorithm is used.[4] The principle behind the method is that just the most probable state sequences should be taken into account. We can quickly locate the ideal tags using the Viterbi algorithm.

Due to the effectiveness of the Viterbi algorithm [Viterbi67] used to decode the NE-class state sequence [7], HMM appears to be utilised in NE recognition more and more frequently.

The following are the HMM Viterbi algorithm's parameters:

**States S in a set where |S|=N.**

The total number of states in this case is N

And Observations O with |O| = k.

The number of output alphabets in this case is k. Probability of Transition, A Emission, B Probabilities of the Initial State

## IV. PROPOSED WORK

The learning by example methodology is applied in the suggested System. It offers a simple process that requires the least amount of work for named entity recognition in any natural language. The person is required to annotate his corpus and test the system for every sentence. Here are the steps to take for any language:

1. Data preparation
2. Estimating Parameters (Training)
3. Evaluate the System

### 4.1 DATA PREPARATION

To make the raw data appropriate for usage in the Hidden Markov model framework for all the languages, we must transform it into trainable form. The training data can be gathered from a variety of sources, including open source, tourism corpora, or even just a plaintext file with a few phrases. Therefore, we must do the following actions in order to transform these files into trainable ones:

---

**Algorithm**

*Step1: Separate each word in the sentence*
*Step2: Tokenize the words*
*Step3: Perform chunking if required*
*Step5: Tag (Named Entity tag) the words by using your experience*
*Step6: Now, the corpus is in trainable form*

**Input:** Raw text file
**Output:** Annotated Text (tagged text)

---

### 4.2 HMM PARAMETER ESTIMATION

Algorithm to estimate the probabilities of various parameters based on states is as below :

---

**Algorithm**

*Step1: Find states.*
*Step2: Calculate Start probability (π).*
*Step3: Calculate transition probability (A)*
*Step4: Calculate emission probability (B)*

**Input**: Annotated tagged corpus
**Output:** HMM parameters

---

#### 4.2.1 PROCEDURE TO FIND STATES

The state is vector contains all the named entity tags candidate interested.

---

**Algorithm**:
*Step1 : For each tag in an annotated text file*
*Step2: If it is already in the state vector*
*Step3: Please ignore it*
*Step4: Otherwise Add to the state vector*

**Input:** Annotated text file
**Output:** State Vector

#### 4.2.2. Procedure to find Start probability

Start probability is the probability that the sentence starts with a particular tag.

$$\text{Start probabilities } (\pi) = \frac{(\textbf{Number of sentences start with particular tag})}{(\textbf{Total Number of Sentences in corpus})}$$

**Algorithm**
*Step1: For each starting tag*
*Step2: Find frequency of that tag as starting tag*
*Step3: Calculate π*

**Input:** Annotated Text file
**Output:** Start Probability Vector

#### 4.2.3. Procedure to find Transition probability

If there is two pair of tags called Ti and Tj, then transition probability is the probability of occurring of tag Tj after Ti.

$$\text{Transition Probability (A)} = \frac{(\textbf{Total Number of sequences from } T_i \textbf{ to } T_j)}{\textbf{Total Number of } T_i}$$

**Algorithm**
*Step1: For each tag in states (Ti)*
*Step2: For each other tag in states (Tj)*
*Step3: If Ti is not equal to Tj*
*Step4: Find frequency of tag sequence Ti Tj, i.e., Tj after Ti*
*Step5: Calculate A = frequency (Ti Tj) / frequency (Ti)*

**Input:** Annotated Text file
**Output:** Transition Probability

#### 4.2.4. Procedure to find emission probability

Emission probability is the probability of assigning a particular tag to the word in the corpus or document.

$$\text{Emission probability (B)} = \frac{\textbf{Total Number of occurance of a word as a Tag}}{\textbf{Total Occurance of that Tag}}$$

**Algorithm**
*Step1: For each unique word, Wi is an annotated corpus*
*Step2: Find frequency of word Wi as a particular tag Ti*
*Step3: Divide frequency by frequency of that tag Ti*

**Input:** Annotated Text file
**Output:** Emission Probability matrix

### 4.3 Implementation Testing

After calculating all these parameters, we apply these parameters to the Viterbi algorithm and testing the sentence as an observation to find named entities.

## V. POS tagging with Hidden Markov Model Implementation

The stochastic method used for POS tagging is called the HMM (Hidden Markov Model). Hidden Markov models are well known for their applications to temporal pattern identification, partial discharges, musical score following, handwriting, gesture recognition, reinforcement learning, and bioinformatics.

For Example: **રમેશ સુરશને જોઇ છે.**

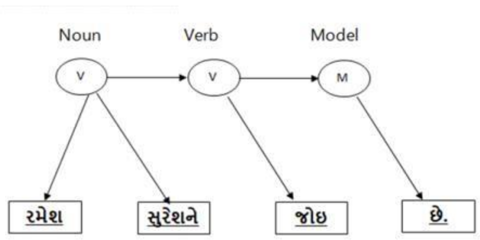



Fig - 2 POS Tagging for Gujarati Sentence

In above Figure - 2, the tags used to determine the likelihood of this specific tag sequence are Noun, Verb, and Model. We must first determine the transition probability and the emission probability.

### 5.1 Transition Probability

The possibility of a specific sequence, such as the likelihood that a noun would be followed by a model, a model by a verb, and a verb by a noun, is known as the transition probability. The Transition Probability is the name given to this likelihood. For a certain sequence to be accurate, it should be high. Let us calculate the above two probabilities for the set of sentences below:

1. રમેશ સુરશને જોઇ છે.
2. સુરશ મહેશને જોઇ છે.
3. શું મહેશ વાાંચે છે ?
4. રમેશ વાાંચે છે.

Table – 1 POS Tagging for Example Sentences

| Example No | Noun | Noun | Verb | Model |
|---|---|---|---|---|
| 1 | રમેશ | સુરશેને | જોઇ | છે. |
| 2 | સુરશે | મહેશને | જોઇ | છે. |
| 3 | શા | મહેશ | વાાંચે | છે. |
| 4 | રમેશ | - | વાાંચે | છે. |

In the above sentences, the word 'રમેશ' appears three times as a noun. To calculate the emission probabilities, Let us similarly create a counting table.

Table – 2 Similarity Table

| Words | Noun | Model | Verb |
|---|---|---|---|
| રમેશ | 2 | 0 | 0 |
| સુરશે | 2 | 0 | 0 |
| મહેશ | 2 | 0 | 0 |
| જોઇ | 0 | 0 | 2 |
| છે | 0 | 4 | 0 |
| વાાંચે | 0 | 0 | 2 |
| શું | 0 | 1 | 0 |

Let's now divide each column by the sum of all of their occurrences. Divide each phrase in the noun column by 6, for instance, since the word "noun" appears six times in the sentences above. After this operation, we obtain the table below.

**Table – 3 Word Occurrences**

| **Words** | **Noun** | **Model** | **Verb** |
|---|---|---|---|
| રમેશ | 2/6 | 0 | 0 |
| સુરશે | 2/6 | 0 | 0 |
| મહેશ | 2/6 | 0 | 0 |
| જોઇ | 0 | 0 | 2/4 |
| છે | 0 | 4/5 | 0 |
| વાાંચે | 0 | 0 | 2/4 |
| શું | 0 | 1/5 | 0 |

From the above table, we infer that

1. The probability that 'રમેશ' is Noun = 2/6 = 0.33
2. The probability that 'સુરશ 'is Model = 2/6 = 0.33

3. The probability that 'મહેશ' is Noun = 2/6 = 0.33
4. The probability that 'જોઇ' is Verb = 2/4 = 0.50
5. The probability that 'વાાંચે' is Verb = 2/4 = 0.50
6. The probability that 'છે' is Verb = 4/5 = 0.80
7. The probability that 'શુાં' is Verb = 1/5 = 0.20

Next, we must calculate the transition probabilities, defining two more tags <S> and <E>. <S> is placed at the beginning of each sentence and <E> at the end, as shown in the figure below.

Table – 4 Transition Probability Calculation

| | | | | | |
|---|---|---|---|---|---|
| **<S>** | **Noun** રમેશ | **Noun** સુરશે ને | **Verb** જોઇ | **Model** છે. | **<E>** |
| **<S>** | Noun સુરશે | Noun મહેશને | Verb જોઇ | Model છે. | **<E>** |
| **<S>** | Model શા | Noun મહેશ | Verb વાાંચે | Model છે. | **<E>** |
| **<S>** | Model - | Noun રમેશ | Verb વાાંચે | Model છે. | **<E>** |

Let us again create a table and fill it with the co-occurrence counts of the tags.

Table – 5 Co-occurrence count of Tag

| <S> | N | M | V | <E> |
|---|---|---|---|---|
| | 3/4 | ¼ | 0 | 0 |
| N | 2/6 | 0 | 4/6 | 0 |
| M | 1/5 | 0 | 0 | 4/5 |
| V | 0 | 4/4 | 0 | 0 |

The <S> tag is followed by the N tag three times, as seen in the above figure. The first entry is therefore 3. The model tag appears just after the <S>. The second entry is therefore 1. The remainder of the table is similarly occupied. Next, determine the likelihood that this sequence is right by using the formula below.

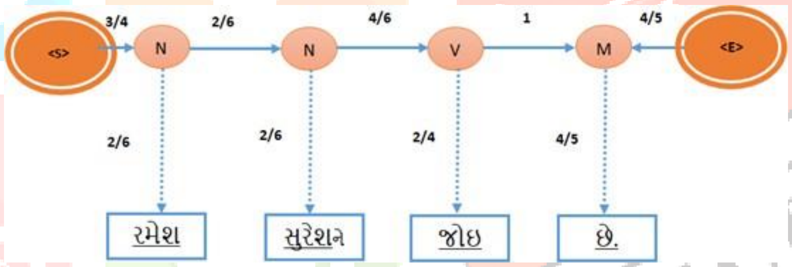



Fig -3 Likelihood of Sequence Tags

The probability of the tag Noun (N) comes after the tag <S> is ¾, as seen in the table. Also, the likelihood that the word જોઇ is a Model is 2/4. In the same manner, we calculate each probability in the graph. Now the product of these probabilities is the likelihood that this sequence is proper. If the tags are not correct, then the product will be zero. Calculating the product of these terms we get,

= 3/4*2/6*2/6*2/6*4/6*2/4*1*4/5*4/5

= 0.00593

Now let us visualize these combinations as paths, and using the transition and emission probability, mark each vertex and edge as shown below.

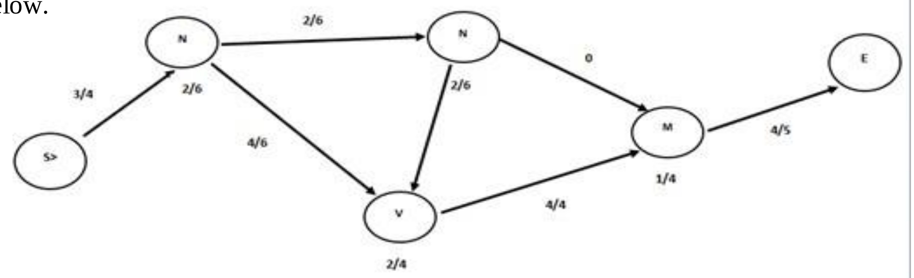



Fig – 4 Emission Probability

Now there are only two paths that lead to the end. Let us calculate the probability associated with each direction.

<S>→N→V→N→M→<E>

=3/4*2/6*4/6*2/4*0*0*1/4*4/5=0.000

<S>→N→N→V→M→<E>

$$=3/4*2/6*2/6*2/6*4/6*2/4*1*4/5*4/5=0.00593$$

The probability of the second sequence is much higher, and hence the HMM will tag each word in the sentence according to this sequence.

## VI. CONCLUSION

Gujarati is the native tongue of Gujarat State and one of the 22 Indian Constitutional Languages.[6] Due to the extensive morphology of Indian languages, it might be challenging to tag words. In this research, I provide a unique method for labelling segments of speech known as the Hidden Markov Model.[1] Sequences of words are tallied by their occurrences using a variety of start probability, transition probability, and emission probability methods. The Hidden Markov model can be used for POS tagging by computing the sequence probability for a text. The proposed approach produces a sequence probability of 0.00593. Therefore, NER systems based on HMM models are quite effective, particularly for Indian languages where there is a lot of variance.

### Declaration

### Authors Contribution

The author makes a substantial contribution to this manuscript. Author himself drafted the manuscript.

### Acknowledgments

The author is grateful to the Department of Computer Science, Vanita Vishram Women's University, Surat, Gujarat for the permission to publish this research.

### Availability of data and material

All relevant data and material are presented in the main paper**.**

### Competing interests

The author declares that they have no competing interests.

### Funding

Not Applicable

## REFERENCES

[1] G. Zhou, "Named Entity Recognition using an HMM-based Chunk Tagger".
[2] D. Shah and H. Bhadka, Paradigm-based morphological analyzer for the Gujarati Language, vol. 989, 2020.
[3] D. Shah and H. Bhadka, Named Entity Recognition from Gujarati Text Using Rule-Based Approach, vol. 736, 2018.
[4] M. C. S, E. Lex and S. Lalitha Devi, "Named Entity Recognition for the Agricultural Domain".
[5] K. Raju Singha, B. Syam Purkayastha and K. Dhiren Singha, "Part of Speech Tagging in Manipuri: A Rule-based Approach," 2012.
[6] A. S. Pillai Professor, "Named Entity Recognition for Indian Languages: A Survey," 2013.
[7] D. N. Mehta and N. P. Desai, "A Survey on Part-Of-Speech Tagging of Indian Languages".
[8] G. Kaur, "Development of Stochastic Part Of Speech Tagger for Morphologically Rich Languages".
[9] N. Aggarwal, S. Amandeep and R. Bgiet, "A Survey on Parts of Speech Tagging for Indian Languages".