

A Trigram HMM Model For Solving Parts-of-Speech (PoS) Tagging Problems

B.S.Uma¹, P.Penchala Prasad²P.G. Student, Department of Computer Science and Engineering, GPREC Engineering College, Kurnool,
Andhrapradesh, India1Assistant Professor, Department of Computer Science and Engineering, GPREC Engineering College, Kurnool,
Andhrapradesh, India2

ABSTRACT: In the most of the Natural language processing problems, we have to model pair of sequences. The Parts Of Speech tagging (PoS) is the best solution for this type of problems. In POS tagging problem, our goal is to build a proper output tagging sequence for a given input sentence. The tag sequence is same as the input sequence. To get the POS tagging we have used the Hidden Markov Model (HMM) along with the Stanford POS parser in this paper.

KEYWORDS: HMM model, PoS Tagging, tagging sequence, Natural Language Processing.

I. INTRODUCTION

In the corpus-linguistics, parts-of-speech tagging (POS) which is also called as grammatical tagging, is the process of marking up a word in the text (corpus) corresponding to a particular part-of-speech based on both the definition and as well as its context.

In the olden days, it is used to be performed by hand, POS tagging is now done in the context of computational linguistics, using algorithms by associating discrete terms as well as hidden parts of speech, in accordance with a set of descriptive tags. POS-tagging algorithms fall into two distinctive groups: rule based and stochastic. Parts-of-speech tagging is harder than just having a set of words and their parts-of-speech, because certain words can be represented as more than one type of speech at the same time. A large percentage of the word-forms are ambiguous. For example, even dogs which are usually thought of as just a plural noun can also be a verb: "The sailor dogs the hatch"

Appropriate grammatical tagging should reflect that dogs used here are Verb, not as plural noun. Analysis is used to infer that "sailor" and "hatch" implicate dogs as action applied to the object "hatch".

II. RELATED WORK

In the childhood we have been taught that there are 9 parts of speech in English: noun, article, adjective, preposition pronoun, adverb, conjunction and interjection. However there are many sub-categories. For nouns, the plural possessive and singular forms can be distinguished. In many languages words are also marked for their "case" (role as subject, object, etc...), grammatical gender, and so on; while verbs are marked for tense, aspect, and so on; while verbs are marked for tense, aspect ,and other things .Linguistics distinguish parts of speech to various fine degrees, reflecting a chosen "tagging system".

In POS tagging the goal is to build a model whose input is a sentence, for example:

"the dog saw a cat"

and whose output is a tag sequence, for example D N V D N (where D represents determiner, N as noun and V as verb).The input to the tagging model is denoted by X_1, X_2, \dots, X_n . It is often referred as a sentence. In the above example the length $n=5$ and $X_1 = \text{the}$, $X_2 = \text{dog}$, $X_3 = \text{saw}$, $X_4 = \text{a}$, $X_5 = \text{cat}$. The output of the tagging model is denoted by Y_1, Y_2, \dots, Y_n . In the above example $Y_1 = D$, $Y_2 = N$, $Y_3 = V$ etc. This type of problem, where the task is to map a sentence to a tag sequence is oft en referred as sequence labeling problem.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015

We will assume that we have a set of training examples, $(X_{(i)}; Y_{(i)})$ for $i=1::: m$, where each $X_{(i)}$ is a sentence and each $Y_{(i)}$ is a tag sequence. Our task is to learn a function that maps sentences to tag sequences from these training examples. To achieve this goal we have used Hidden Markov Model (HMM) for this alignment process.

III. PROPOSED METHOD

The proposed method gives an approach of finding different parts-of-speech for a given input sequence. This is achieved by using

- 1) Trigram HMM model
- 2) Stanford parser.

Definition of trigram HMM

A trigram HMM consists of a finite set of V possible words, and a finite set K of possible tags, together with the following parameters:

- A parameter $q(s/u,v)$ for any trigram u,v,s such that $s \in k \cup \{STOP\}$ and $u,v \in V \cup \{*\}$. The value for $q(s/u,v)$, can be interpreted as the probability of seeing the tags immediately after the bigrams of u,v .
- A parameter $e(x/s)$ for any $x \in V, s \in K$. The value for $e(x/s)$ can be interpreted as the probability of seeing observation x paired with state s .

Define S to be the set of all sequence / tag-sequence pairs $(x_1, \dots, x_n, y_1, \dots, y_n)$ such that $n \geq 0, x_i \in V$ for $i=1, \dots, n$ and $y_i \in K$ for $i=1, \dots, n$ and $y_n = STOP$.

We then define the probability for any $(x_1, \dots, x_n, y_1, \dots, y_n) \in S$ a $P(x_1, \dots, x_n, y_1, \dots, y_n) = P$ which is given by

$$\prod_{i=1}^n q(y_i | y_{i-1}, y_i) \prod_{i=1}^n e(x_i | y_i)$$

As an example if we have $n=3, x_1 \dots x_3$ equal to the sentence *the dog laughs* and Y_1, \dots, Y_4 equal to the tag sequence $D N V STOP$, then $P(x_1, \dots, x_n, y_1, \dots, y_{n+1}) = q(D | *, *) \times q(N | *, D) \times q(V | D, N) \times q(STOP | N, V) \times e(the | D) \times e(dog | N) \times e(laughs | V)$

Independence Assumptions in Trigram HMMs

Consider a pair of sequences of random variables X_1, \dots, X_n , and Y_1, \dots, Y_n , where n is the length of sequences. We assume that each X_i can take any value in a finite set V of words. For example, V might be a set of possible words in English, for example $V = \{the, dog, saw, cat, laughs, \dots\}$ Each Y_i can take any value in a finite set K of possible tags. For example, K might be the set of possible Part-Of-Speech tags for English, e.g. $K = \{D, N, V, \dots\}$

The length n is itself a random variable – it can vary across different sentences but we will use a similar technique to the method used for modeling variable length Markov process.

Our task will be to model the joint probability $P(X_1 = x_1, \dots, X_n = x_n, Y_1 = y_1, \dots, Y_n = y_n)$ for any observation sequence x_1, \dots, x_n paired with a state sequence y_1, \dots, y_n for any observation sequence x_1, \dots, x_n paired with a state sequence y_1, \dots, y_n , where each x_i is a member of V and each y_i is a member of K .

The following process is a stochastic one which generates sequence pairs $y_1, \dots, y_{n+1}, x_1, \dots, x_n$:

1. Initialize $i=1$ and $y_0 = y_{-1} = *$.
2. Generate y_i from the distribution $q(y_i | y_{i-2}, y_{i-1})$

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015

3. If $y_i = \text{STOP}$ then return $y_1 \dots y_i, x_1 \dots x_{i-1}$. Otherwise, generate x_i from the distribution $e(x_i|y_i)$, set $i = i+1$, and return to step 2.

Parameters of Trigram HMM

With the accessed training data which is containing a set of examples, where each example is a sentence $x_1 \dots x_n$ paired with tag sequence $y_1 \dots y_n$. With these data we have estimated the parameters in the following way:

Define $C(u, v, s)$ to be the number of times the sequence of three states (u, v, s) is seen in training data: for example, $C(V, D, N)$ would be the number of times the sequence of three tags V, D, N is seen in the training corpus. Similarly, define $C(u, v)$ to be the number of times the tag bigram (u, v) is seen. Define $C(s)$ to be the number of times that the state s is seen in the corpus. Finally define $C(s \rightarrow x)$ to be the number of times that the state s is seen paired with the observation x in the corpus: for example $C(N \rightarrow \text{dog})$ would be the number of times dog is seen paired with the tag N .

Given these definitions the maximum-likelihood estimates are

$$q(s|u, v) = \frac{c(u, v, s)}{c(u, v)} \quad \text{and}$$

$$e(x/s) = \frac{c(s \rightarrow x)}{c(s)}$$

For example, we would have the estimates

$$q(N | V, D) = \frac{c(V, D, N)}{c(V, D)}$$

And
$$e(\text{dog}/N) = \frac{c(N \rightarrow \text{dog})}{c(N)}$$

Thus estimating the parameters of the model is simple, just read off counts from the training corpus, and then compute the maximum - likelihood estimates.

Decoding with HMMs: viterbi Algorithm

The main problem lies in finding the most appropriate tag sequence for an input sentence. This is the problem of finding

$$\arg \max_{y_1 \dots y_{n+1}} p(x_1 \dots x_n, y_1 \dots y_{n+1})$$

Where the argmax is taken over all sequences $y_1 \dots y_{n+1}$ such that $y_i \in K$ for $i = 1 \dots n$ and $y_{n+1} = \text{STOP}$.

The naive brute force method would simply enumerate all possible tag sequences $y_1 \dots y_{n+1}$, score them under a function p , and takes the highest scoring sequence. For example, given the input sentence

the baby crawls

and assuming the set of possible tags is $K = \{ D, N, V \}$, we have to consider all possible tag sequences

D D D STOP

D D N STOP

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015

D D V STOP
D N D STOP
D N N STOP
D N V STOP

.....

There are $3^3 = 27$ possible sequences in this case. However this method is inefficient for longer sentences. For an input sentence of length n , there are $|K|^n$ possible tag sequences. The exponential growth with respect to the length n means that for any reasonable length sentence brute force search will not be tractable.

Instead we can efficiently find the highest probability tag sequence using a dynamic programming algorithm called viterbi algorithm. The input to the algorithm is a sentence $x_1 \dots x_n$. Given this sentence, for any $K \in \{1, \dots, n\}$, for any sequence $y_1 \dots y_k$ such that $y_i \in K$ for $i = 1 \dots k$, the function is defined as

$$r(y_1 \dots y_k) = \prod_{i=1}^k q(y_i | y_{i-2}, y_{i-1}) \prod_{i=1}^k e(x_i | y_i)$$

The basic algorithm can be defined as of follows

Input: a sentence $x_1 \dots x_n$ parameters $q(s|u,v)$ and $e(x|s)$

Initialization: Set $\pi(0,*,*) = 1$ and $\pi(0, u, v) = 0$ for all (u,v) such that $u \neq *$ or $v \neq *$

Algorithm:

- For $K=1$ to n
 - For $u \in K, v \in K$,
 - $\Pi(k,u,v) = \max_{w \in K} (\pi(k-1, w, u) * q(v|w, u) * e(x_k|v))$
- Return $\max_{u \in K, v \in K} (\pi(n, u, v) * q(\text{STOP}|u, v))$

ALGORITHM1: Basic Vitebri Algorithm

The parts-of-speech values in this project are obtained by using an open source tool Stanford parser, which we have trained with our own models. To do this, the tagger has to load a “trained” file that contains the necessary information for the tagger to tag the string. This “trained” file is called a model and has the extension “.tagger”.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015

IV. EXPERIMENTAL RESULTS

Figures show the results of word alignment from a sentence and PoS tagging by using HMM model with vitebri algorithm.

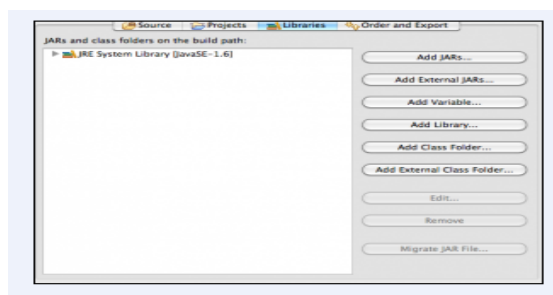


FIG1: Adding Jar files to Parser

The above figure shows the procedure of adding the java code into the Stanford Parser. The JAR files are included by adding the external archive files where the .java file is located in the system.

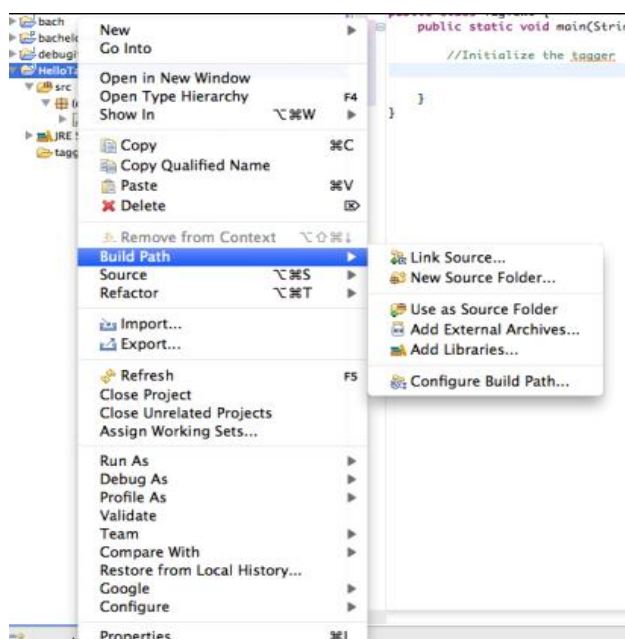


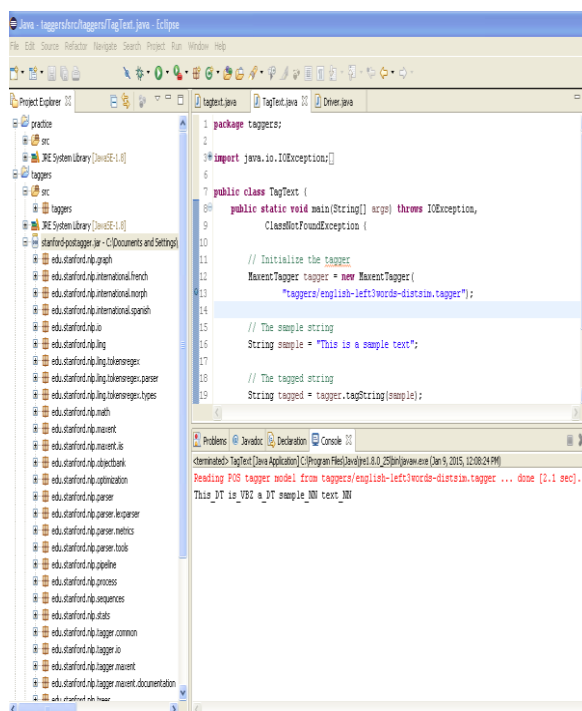
FIG2: Importing model file

In order to include PoS tagger we have to include the .model file, where the classes in the model file represents various languages containing the taggers extracting the speech of the text as per the context.

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2015



```

package taggers;
import java.io.IOException;
classNotFoundExcep...
public class TagText {
    public static void main(String[] args) throws IOException,
classNotFoundExcep...
// Initialize the tagger
MaxentTagger tagger = new MaxentTagger(
"taggers/english-left1words-diststm.tagger");
// The sample string
String sample = "This is a sample text";
// The tagged string
String tagged = tagger.tagString(sample);

```

Problems | Javadoc | Declaration | Console

terminated: TagText [Java Application] [C:\Program Files\Java\jdk1.8.0_251\bin\java.exe [Jan 5, 2015, 12:08:24 PM]

Reading POS tagger model from taggers/english-left1words-diststm.tagger ... done [2.1 sec].

This JT is VBI a JT sample NN text NN

FIG3: various parts-of-speech as output

The above figure contains DT determiner, NN noun representing the PoS in various forms in standard format as represented by Stanford dependencies.

V. CONCLUSION

We have implemented an automatic PoS detection technique from various inputs. Our algorithm successfully detects the matching output sequence from the tagging input sequences which consists of mixed textual content. We have applied our algorithm on many inputs and found that it successfully detect the matching output sequence.

REFERENCES

- [1]Brants,T.(2000).A Statistical Part-of-Speech Tagger. Sixth Applied Natural Language Processing Conference.
- [2]Jurafsky,D.,& Martin,J.H.(2008).Speech and Language Processing. Prentice Hall
- [3]J.Och and H.Ney.2003.A systematic comparison of various statistical alignment models. Computational Linguistics, 29(1):19-51, March.
- [4]Weischedel,R.,Schwartz,M.,&Ramshaw,R.(1993). Coping with Ambiguity and unknown words through probabilistic Models.
- [5]D.Melamed.2000.Models of translational equivalence among words.Computational Linguistics,26(2):221-249
- [6]Toutanova,H.T.Ilhan, and C.D.Manning, 2002. Extensions to hmm based statistical word alignment models.In Proc.conf.on Empirical Methods of Natural Language Processing,pages 87-94, philadelphia, PA.