Bigram HMM Tagger

POS Tagger

Shobhika Panda (*Author*)

Dept. of Computer Science

UT DALLAS

Texas, USA

sxp150031@utdallas.edu

***Abstract*—This paper presents the following ideas: i) Usage of probabilities(HMM) to do training ii) Bayesian inference model is the inspiration of this project iii) Choosing the most probable tags iv) Using Viterbi to decode a new test sentence. Combining all of these, the tagger gives 93.24% of accuracy on WSJ data and error rate of 6% when run with Viterbi Algorithm and 91% of accuracy and error rate of 8 % when run with Baseline Algorithm.**

Keywords—WSJ – Wall Street Journal Corpus, Viterbi, HMM

Introduction

POS tagging is an extremely important topic in statistical Natural Language processing. The fact that POS tagging is an important component of NLP is well recognized but the computation task of POS gained visibility in recent time in last 10 to 15 years. It is well understood that a good POS tagger is very important in NLP task. Before beginning any NLP processing on text it is important to know the categories of the words that form the text. This paper will describe the POS tagging problem and the statistical approach to do this. POS is best done by using machine learning techniques on annotated corpora. Human created rules for POS tagging which do a good job but they are brittle and subject to human error and may miss out phenomena. But on the other hand if we have data which is already marked with POS tags and we run machine learning technique on this then it is possible to create a statistical system which does a very good job of POS tagging. POS is a process which attaches each word in a sentence with a suitable tag from a given set of tags. The set of tags is called the tag set. Here are some examples of POS Tags:

NN- Noun; e.g. Dog

VB – Verb; e.g. Eat

JJ – Adjective; e.g. Red

PRP – Pronoun; e.g. You

NNP – Proper Noun; e.g. Jane

A. Problems in POS Tagging

POS Tag ambiguity is very common in text. For example, I bank on the bank on the river bank for my transactions. Bank1 is the verb and the other two banks are noun.

B. *Process*:

For each words, we list the all possible tag for each word in the sentence.

Choose the best suitable tag sequence.

We are interested in finding the best possible path from start to dot in each sentence if we visualize this as a graph, if there are n words/observations in a sentence, and states are considered as tags.

C. Challenges in POS Tagging

For any set of data, we have unseen data which is not seen in limited training set. It is important to develop methods that deal with words that are not seen.

D. Handling Challenges

It is always good to see in terms of statistical preferences, how likely a word is verb or noun and what is most probable bigram sequence for eg. Det N(the follow tag) occurs very frequently.

I. RELATED PROJECTS

A. Brown Corpus History

Research on POS tagging has been going on from mid-1960's. the first corpus was developed at Brown University by Henry Kucera and W.Nelson Francis. It maintains millions of words of English text. The brown corpus is tagged with POS tags since historical times. The first enhancement was done by Greene and Rubin, which had a list of handmade tags that can exist at the same time. The experiment was about 70% true. The results were iteratively made right, so by 1970's the tagging got better.

B. Efficient Methods

In recent times, the methods of POS tagging have taken leaps and bounds. Fine grained techniques of unknown words have been incorporated, a proper usage of previous and follow tags through a dependency representation has also been seen getting followed. Some researchers are hell bent on taking the performance from 90% to 100% accuracy. Stanford and Penn state have been wonderful contributors to open source technologies in NLP. Some have moved from statistical approaches to inference engines by using logic theory.

II. APPROACH

HMM Model is a powerful tool for statistical NLP. POS tags are learnt by training from corpus data. The machine which is used is HMM model. NLP happens at many layers. Many tasks are sequence labelling tasks which are carried out in layers. Within a layer there might be limited information, we must be resilient to uncertainty. Probability and Markov model is the solution to our problem.

Motivation: Bayes Inference

The problem at hand is predicting the state sequence from the observation sequence. State sequence is tag sequence and observation sequence is word sequence.

Three main constituents of HMM are:

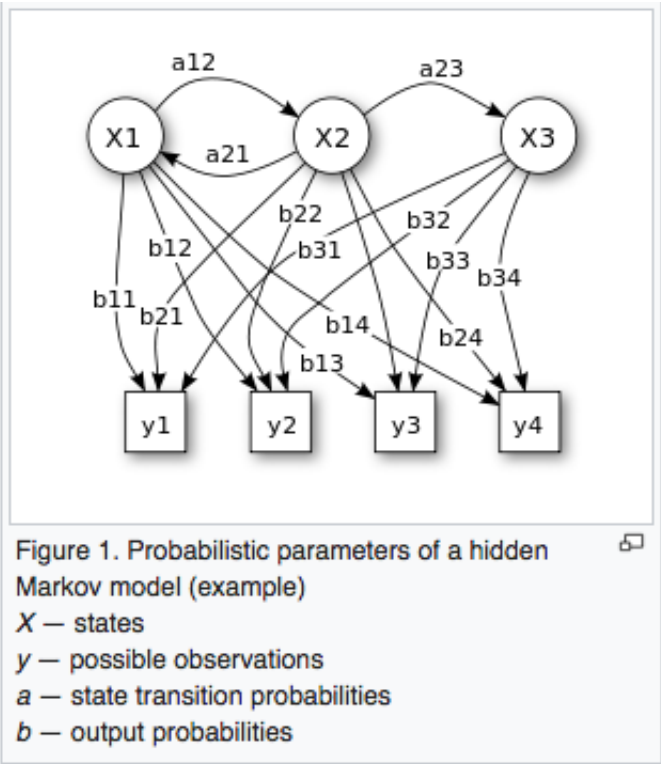


Figure 1. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities

Initial State Probability: $P(S_i)$, initial probability for each state i.

Transition Probability: $P(S_j | S_i)$ transition from state i to j.

Emission Probability: $P(O_j|S_i)$ probability of observation given state i.

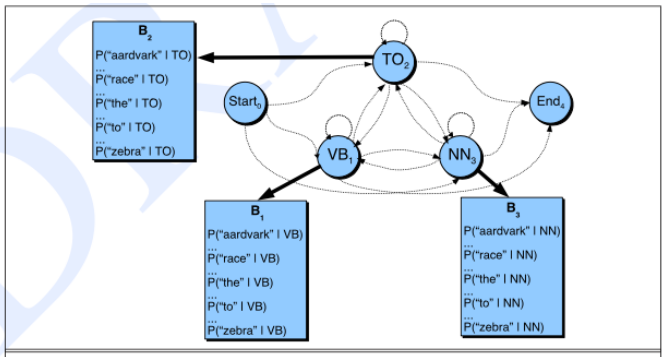


Figure 5.14 The *B* observation likelihoods for the HMM in the previous figure. Each state (except the non-emitting Start and End states) is associated with a vector of probabilities, one likelihood for each possible observation word.

Goal:

Maximize P(S|O) where S is State Sequence and O is observation sequence.

S* = argmax $_s$ P(S|O), we vary S and we obtain different probability values, where we reach maximum of P, we record that particular S, output is given which is best possible S.

Baye's Theorem:

P(A|B) = P(A)*P(B|A)/P(B)

P(A): Prior

P(B|A): Likelihood

argmax_s P(S|O) = argmax_s P(S)*P(O|S), P(O) denominator can be ignored since it is constant factor and is independent of S.

How can we actually use HMM model to predict POS Tags?

Viterbi Algorithm:

HMM holds hidden variables, the task of determining which sequence of variables is the hotspot for some succession of observations will be called the decoding task.

Viterbi is the most common deciphering algorithm utilized for HMM's if it is part of speech tagging or speech recognition.

Viterbi takes HMM as input and set of observed words *O* = (*o*1*o*2*o*3 ...*oT*) and returns the most probable tag sequence *Q* = (*q*1 *q*2 *q*3 . . . *qT)*

We are looking at the equation which summarizes Viterbi as a whole.

A. Equations

$$v_t(j) = \max v_{t-1}(i) a_{ij}\, b_j(o_t) \qquad (1)$$

The three factors multiplied in equation(1) for extending the previous paths to compute Viterbi probability at time t are:

$V_{t-1}(i)$: the previous Viterbi path probability from the previous time step.

a_{ij}: the transition probability from previous state q_i to current state q_j.

$b_j(o_t)$: the state observation likelihood of the observation o_t given current state j

B. Algorithm

function VITERBI(*observations* of len *T*,*state-graph* of len *N*) **returns** *best-path*

create a path probability matrix *viterbi*[*N*+2,*T*]
for each state *s* **from** 1 **to** *N* **do** ;initialization step
  viterbi[*s*,1] ← $a_{0,s} * b_s(o_1)$
  backpointer[s,1] ← 0
for each time step *t* **from** 2 **to** *T* **do** ;recursion step
  for each state *s* **from** 1 **to** *N* **do**
    viterbi[s,t] ← $\max_{s'=1}^{N}$ *viterbi*$[s',t-1] * a_{s',s} * b_s(o_t)$
    backpointer[s,t] ← $\text{argmax}_{s'=1}^{N}$ *viterbi*$[s',t-1] * a_{s',s}$
viterbi[q_F,T] ← $\max_{s=1}^{N}$ *viterbi*$[s,T] * a_{s,q_F}$; termination step
backpointer[q_F,T] ← $\text{argmax}_{s=1}^{N}$ *viterbi*$[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from *backpointer*[q_F,*T*]

Figure 5.17 Viterbi algorithm for finding optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state-path through the HMM which assigns maximum likelihood to the observation sequence. Note that states 0 and q_F are non-emitting.

III. IMPLEMENTATION

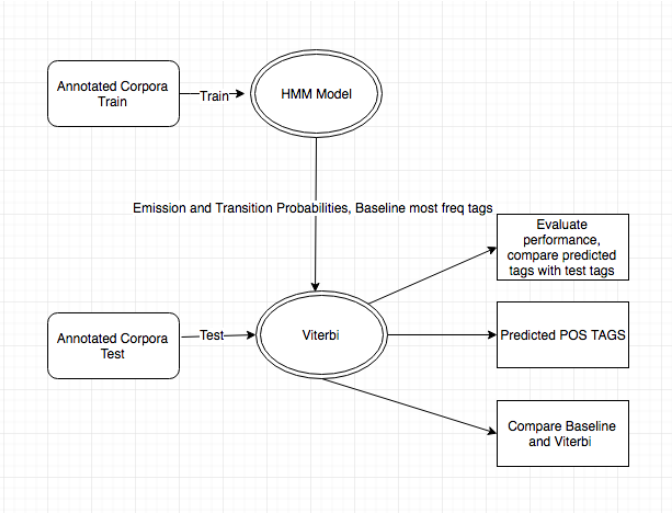
The implementation part was very crucial to understand how HMM and Viterbi works. First the annotated data was trained using HMM Model. Nested Hash map was used in the implementation. To store the transition probabilities, outer key was the train tag and inner Hash map was the value, and inner Hash map key was the follow tag and the value was the frequency of occurrence of follow tag following the current tag. Another nested Hash map was used to store emission probabilities, where each word was key in the outer Hash map and value was the inner Hash map. Inner Hash map key was tag

of the word and value was the frequency of occurrence of tag with the same word. Another Hash map was made for word tag baseline, which contains word as key and most frequent tag as value. Since the start and end tag is not mentioned. In this implementation, "." has been accounted as start and end tag for a sentence. The first sentence doesn't have a dot so I have made necessary changes to adjust the same.

Second step of HMM included summing up all counts and diving each count by sum to obtain the probabilities. I used a Tree map so that sorting wouldn't be needed, Tree map does that.

Third step is to pass on the probabilities and baseline map obtained to the Viterbi. Viterbi reads the test file with words and tags. It takes into account previous path probabilities, calculates the max probability and simultaneously predicts tags and checks it with test tags, if they are not equal then it counts the number of errors in Viterbi and same for baseline. 1-error gives the accuracy.

A. Architecture



B. Data Source

The annotation of the data has been derived from WSJ Corpus by a program written by Sabine Buchholz from Tilburg University, The NetherLands. The train and test data consists of three columns separated by spaces. Each word has been put on a separate line and there is an empty line after each sentence. The first column contains the current word, the second its part-of-speech tag as derived by the Brill tagger and the third its

of-speech as derived by Brill Tagger and third its chunk tag as derived from the WSJ corpus. We wont be needing the third column. In this implementation only first two columns have been taken into account. Another data set from WSJ has also been taken into account for testing. In this data set words are placed separately in each line separated by "/" and consisting of part-of-speech tags on the other side of "/".

C. Figures and Tables

WSJ Data Set 1:

Train Tokens
211727

Test Tokens
47377

WSJ Data Set 2:

Train Tokens
395

Test Tokens
13

WSJ DATA SET 1 :TABLE(1)

MODEL	NO OF ERROS	FRACTION OF ERROS	ACCURACY
BASELINE	3924	0.082	91.717
VITERBI	3201	0.067	93.24

WSJ DATA SET 2: TABLE(2)

MODEL	NO OF ERROS	FRACTION OF ERROS	ACCURACY
BASELINE	1	0.0769	92.3
VITERBI	1	0.0769	92.3

First 7 TAGS DATASET2 (TABLE 3)

Tags Suggested By Baseline	Tags Suggested By Viterbi	Correct Tags
NNP	NNP	NNP
VBD	VBD	VBD
IN	IN	WDT
VBD	VBD	VBD
VBN	VBN	VBN
TO	TO	TO
NNS	NNS	NNS

TABLES SHOWING ERRONEOUS COUNT FOR DATASET 1 (TABLE 4)

Actual Tags	Predicted Tags	Count of erroneous predicted Tag
JJ	NN	20
JJ	NNP	13
JJ	RB	1
JJ	VB	2
NN	IN	3
NN	JJ	91
NN	MD	1
NN	NNP	6
NN	VB	20
NN	VBG	8
NN	VBP	1
PDT	DT	4
PDT	NN	6
VBN	JJ	7
VBN	NN	4
VBN	VB	6
VBN	VBD	634
RBR	JJ	1
RBR	JJR	70
VBP	NN	145
VBP	VB	206
VBP	VBN	5
IN	DT	6
WDT	IN	95
NNPS	NNP	46
VB	IN	4
VB	JJ	26
VB	NN	202
VB	VBN	3
VB	VBP	7
VBZ	NNS	461
VBZ	POS	6
PRP	CD	1
RB	DT	9
RB	EX	2
RB	IN	108
RB	JJ	245
RB	NN	6
RB	NNP	1
DT	IN	22
NNP	JJ	209
NNP	NN	165
NNP	NNPS	3
NNS	NNP	16
NNS	NNPS	2
NNS	VBZ	3
PRP$	PRP	10
POS	"	33
POS	NNS	8
VBD	VB	1

VBD	VBN	36
VBG	NN	199
RP	IN	8
RP	RB	4

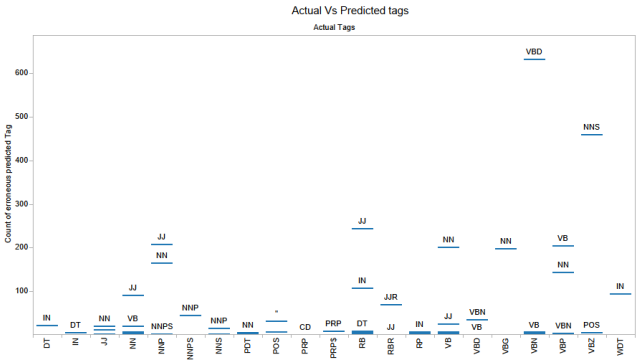
Table 4 shows predicted tags against actual tag and the count of each predicted tag, showing how many times the error occurred. Here we notice that JJ gets tagged most of the times as Noun and VBN as VBD, which shows that the model is not that strong in differentiating well.

This is the Gantt View Chart made from tableau:

X Axis: Actual Tags

Y Axis: Counts of predicted tags

X space area: Predicted tags with actual counts, for each of the actual tags.



IV. CONCLUSION

The conclusions that can be derived from the results and erroneous results:

- If the test data is really less in size like the dataset2 it works really well with less number of erroneous results.
- The accuracy is still great with 93% in dataset1 and error rate is 6%. Viterbi performs better than baseline which is based on most frequent tags.
- VBN is tagged as VBD most of the times, 634 to be exact from dataset1.
- Noun is tagged as JJ 209 times.
- Some unknown words like proper names since they have never been seen, do not get tagged that well.
- VBZ is tagged as NNS most of the times.

V. FUTURE WORK

- Extending the HMM Algorithm to Trigrams.
- To achieve high accuracy with part-of-speech taggers, it is also important to have a good model for dealing with unknown words.
- Proper names and acronyms are unknown words created very often, and even new common nouns and verbs enter the language at a surprising rate. One useful feature for distinguishing parts of speech is word shape: words starting with capital letters are likely to be proper nouns (NNP).
- The best possible way to predict part-of-speech tags for unknown words is morphology.
- For example, words that end in *-s* are likely to be plural nouns (NNS), words ending with *-ed* tend to be past participles (VBN), words ending with *able* tend to be adjectives (JJ), and so on.
- Most recent approaches to unknown word handling, however, combine these features in a third way: by using maximum entropy models such as the **Maximum Entropy Markov Model** (**MEMM**).

VI. REFERENCES

[1] Introduction to Natural Language Processing by **Jufrasky**

[2] http://www.hlt.utdallas.edu/~moldovan/CS6320.17S/notes.html

[3] Tableau for the chart

[4] Eclipse for Java.

[5] Data Source: http://www.cnts.ua.ac.be/conll2000/chunking/.