POS Tagging Using Hidden Markov Model

Prashant Prakash

Natural Language Processing

The University of Texas at Dallas

Richardson, TX 75080 USA

pxp141730@utdallas.edu

Abstract

POS Tagging is one of the important feature in Natural Language Processing. POS Tagging is about identifying Part of Speech of words in the sentence. For a System to understand the sense of a sentence Part of speech Tagging is really required. This report aims at doing Part of speech tagging using Hidden Markov Model and further Viterbi Algorithm is used to optimize the complexity of tagging words. Sentences from Brown Corpus are picked for training and testing the HMM model.

1 Introduction

In Corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up a word in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context—i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph. A simplified form of this is commonly taught to school-age children, in the identification of words as nouns, verbs, adjectives, adverbs, etc.

Research on part-of-speech tagging has been closely tied to corpus linguistics. The first major corpus of English for computer analysis was the

Brown Corpus developed at Brown University by Henry Kučera and W. Nelson Francis, in the mid-1960s. It consists of about 1,000,000 words of running English prose text, made up of 500 samples from randomly chosen publications. Each sample is 2,000 or more words (ending at the first sentence-end after 2,000 words, so that the corpus contains only complete sentences).

The Brown Corpus was painstakingly "tagged" with part-of-speech markers over many years. A first approximation was done with a program by Greene and Rubin, which consisted of a huge handmade list of what categories could co-occur at all. For example, article then noun can occur, but article verb (arguably) cannot. The program got about 70% correct. Its results were repeatedly reviewed and corrected by hand, and later users sent in errata, so that by the late 70s the tagging was nearly perfect (allowing for some cases on which even human speakers might not agree).

For some time, part-of-speech tagging was considered an inseparable part of natural language processing, because there are certain cases where the correct part of speech cannot be decided without understanding the semantics or even the pragmatics of the context. This is extremely expensive, especially because analyzing the higher levels is much harder when multiple part-of-speech possibilities must be considered for each word.

2 Related Work

In the mid 1980s, researchers in Europe began to use hidden Markov models (HMMs) to disambiguate parts of speech, when working to tag the Lancaster-Oslo-Bergen Corpus of British English. HMMs involve counting cases (such as from the Brown Corpus), and making a table of the probabilities of certain sequences. For example, once you've seen an article such as 'the', perhaps the next word is a noun 40% of the time, an adjective 40%, and a number 20%. Knowing this, a program can decide that "can" in "the can" is far more likely to be a noun than a verb or a modal. The same method ca n of course be used to benefit from knowledge about following words.

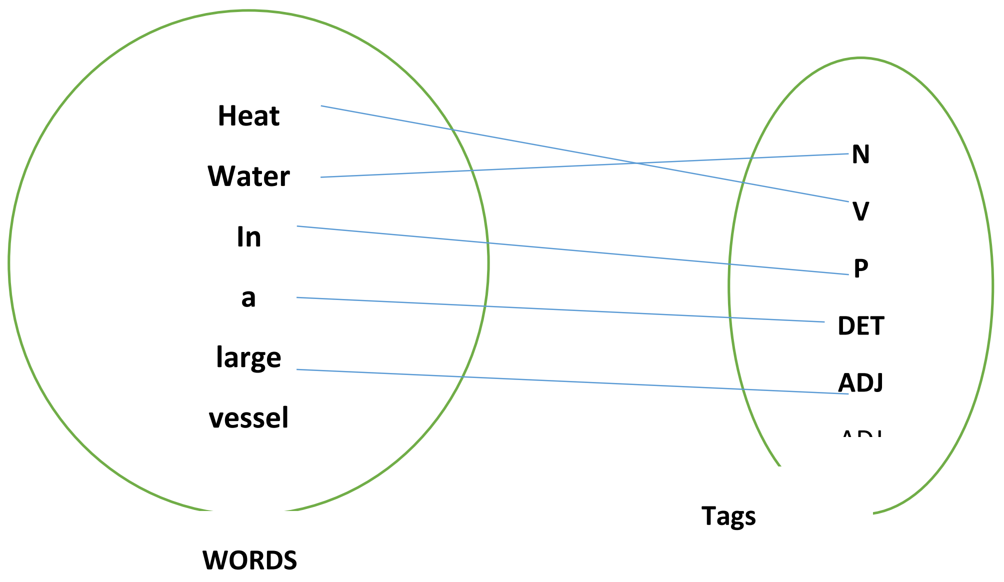
More advanced ("higher order") HMMs learn the probabilities not only of pairs, but triples or even larger sequences. So, for example, if you've just seen a noun followed by a verb, the next item may be very likely a preposition, article, or noun, but much less likely another verb.

Stanford University is one of the major contributor in providing data sets for POS tagging. They have

POS tagger software available in multiple languages. Also Stanford University is carrying multiple research on this and trying to improve the accuracy.

3 POS Tagging

POS tagging of assigning a part-of-speech to each word in a sentence. (automatically or manually). It is useful in resolving lexical ambiguity.



3.1 Why Useful

POS tagging is useful in

- Information Retrieval
- Text to Speech: object(N) vs. object(V)
- Word Sense Disambiguation

POS tagging is useful as a preprocessing step of parsing.

- Unique tag to each word reduces the number of parses.

3.2 Why POS tagging is hard?

POS tagging is hard mainly because of "**Ambiguity**". The major ambiguity found in tagging a word to its part of speech is "Noun" gets tagged to Verb most of the times and vice-versa. Few examples are shown below:

I. "Plants/N need light and water."
II. "Each one plant/V one."
III. "Flies like a flower"
    a. Flies: noun or verb?
    b. like: preposition, adverb, conjunction, noun, or verb?
    c. a: article, noun, or preposition?
    d. flower: noun or verb?

3.3 Methods for POS tagging

There are different approaches taken to do POS tagging.

- Stochastic tagging: Maximum likelihood, Hidden Markov model tagging.
  Pr(Det-N) > Pr (Det-Det)
- Rule based tagging: if <some pattern> then … <some part of speech > (allows for several passes)
- Constrain Grammar (CG) tagging: The cautious approach: ‘Don’t guess, just eliminate the impossible!’
- Transmation-based (TB) tagging: The whimsical approach: ‘ Guess first , then change your mind is necessary!’

4 Hidden Markov Model

A hidden Markov model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states.

In simpler Markov models (like a Markov chain), the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but output, dependent on the state, is visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by an HMM gives some information about the sequence of states. The adjective 'hidden' refers to the state sequence through which the model passes, not to the parameters of the model; the model is still referred to as a 'hidden' Markov model even if these parameters are known exactly.

Hidden Markov Model consists of three components:

I. $Ps(Si)$: Probability of system starting in state Si
II. $Ps(Sj|Si)$: Probability of the system transitioning from state Si to Sj.
III. $Pe(Xj|Si)$: Probability of the system emitting output Xj in state Si.

4.1 Architecture

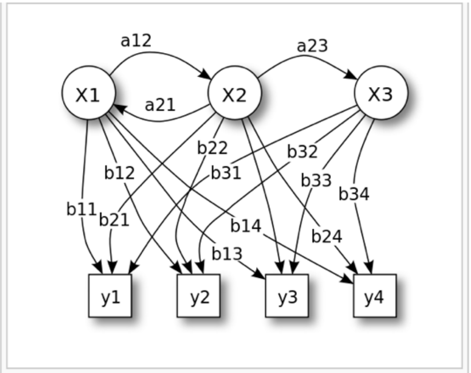
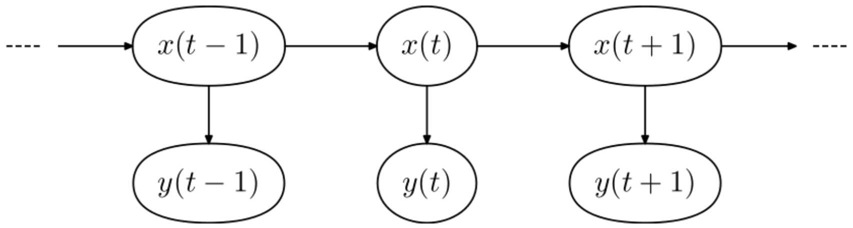


Figure 1. Probabilistic parameters of a hidden Markov model (example)
X — states
y — possible observations
a — state transition probabilities
b — output probabilities

The diagram below shows the general architecture of an instantiated HMM. Each oval shape represents a random variable that can adopt any of a number of values. The random variable x(t) is the hidden state at time t (with the model from the above diagram, x(t) ∈ { x1, x2, x3 }). The random variable y(t) is the observation at time t (with y(t) ∈ { y1, y2, y3, y4 }). The arrows in the diagram (often called a trellis diagram) denote conditional dependencies.



From the diagram, it is clear that the conditional probability distribution of the hidden variable x(t) at time t, given the values of the hidden variable x at all times, depends only on the value of the hidden variable x(t − 1); the values at time t − 2 and before have no influence. This is called the Markov property. Similarly, the value of the observed variable y(t) only depends on the value of the hidden variable x(t) (both at time t).

4.2 A Concrete Example

Consider two friends, Alice and Bob, who live far apart from each other and who talk together daily over the telephone about what they did that day. Bob is only interested in three activities: walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day. Alice has no definite information about the weather where Bob lives, but she knows general trends. Based on what Bob tells her he did each day, Alice tries to guess what the weather must have been like.

Alice believes that the weather operates as a discrete Markov chain. There are two states, "Rainy" and "Sunny", but she cannot observe them directly, that is, they are hidden from her. On each day, there is a certain chance that Bob will perform one of the following activities, depending on the weather: "walk", "shop", or "clean". Since Bob tells Alice about his activities, those are the observations. The entire system is that of a hidden Markov model (HMM).

Alice knows the general weather trends in the area, and what Bob likes to do on average.

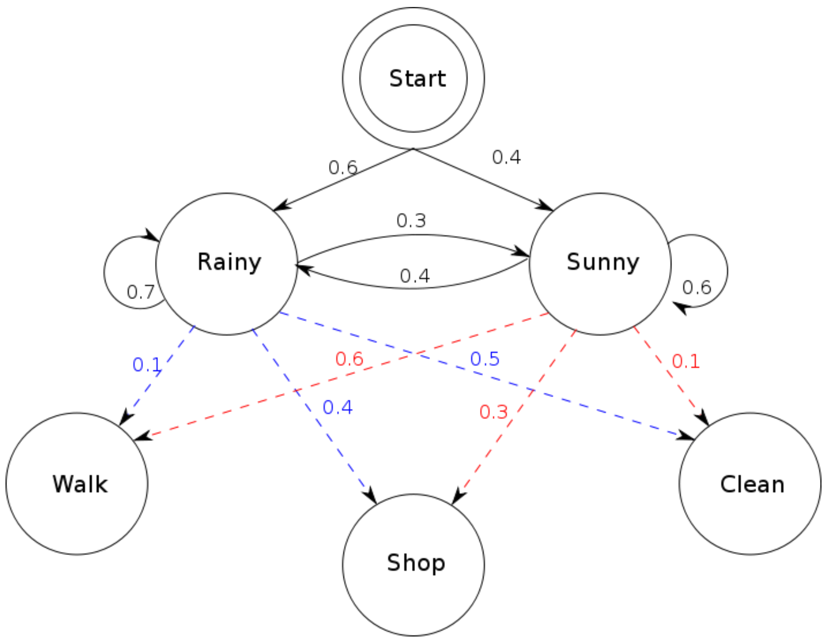
```
states = ('Rainy', 'Sunny')

observations = ('walk', 'shop', 'clean')

start_probability = {'Rainy': 0.6, 'Sunny': 0.4}

transition_probability = {
   'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
   'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6},
   }

emission_probability = {
   'Rainy' : {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},
   'Sunny' : {'walk': 0.6, 'shop': 0.3, 'clean': 0.1},
   }
```



In this piece of code, start_probability represents Alice's belief about which state the HMM is in when Bob first calls her (all she knows is that it tends to be rainy on average). The particular probability distribution used here is not the equilibrium one, which is (given the transition probabilities) approximately {'Rainy': 0.57, 'Sunny': 0.43}. The transition_probability represents the change of the weather in the underlying Markov chain. In this example, there is only a 30% chance that tomorrow will be sunny if today is rainy. The emission_probability represents how likely Bob is to perform a certain activity on each day. If it is rainy, there is a 50% chance that he is cleaning his apartment; if it is sunny, there is a 60% chance that he is outside for a walk.

4.3 Complexity

5 Viterbi Algorithm

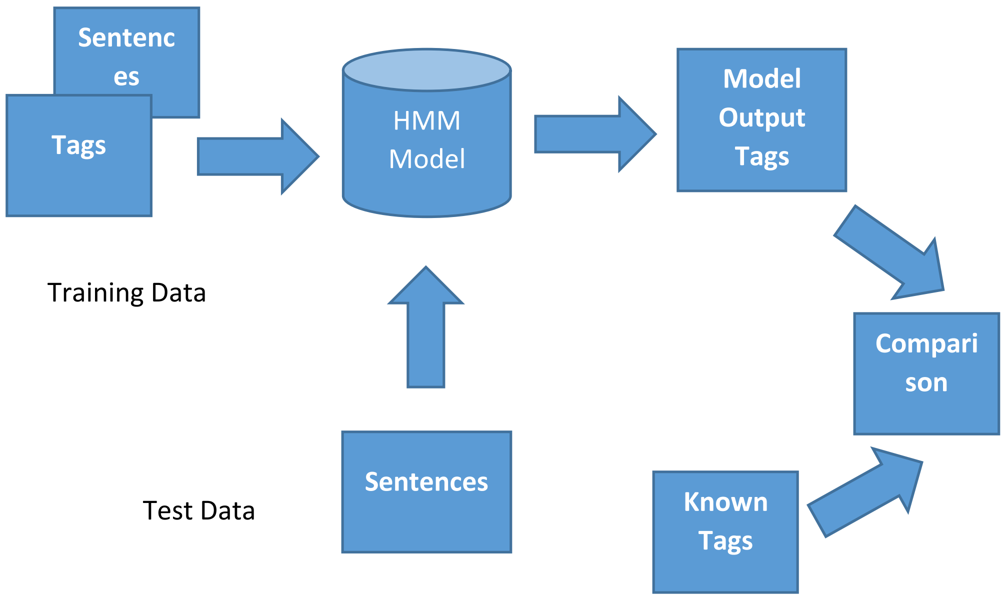
Suppose that our corpus is a k-tag Treebank with tags t1,t2,…,tk and m words w1,w2,…,wm is the dictionary. Let $P[r,s]$ for $1 \le r \le n\ , 1 \le s \le k$ be the greatest probability among all probabilities of sequence of tags $T1T2 \ldots Tr\ with\ Tr = ts$.

Let $L[r,s]$ for $1 \le r \le n\ , 1 \le s \le k$ be the sequence of tags $T1T2 \ldots Tr\ with\ Tr = ts$ corresponding to that probability. Then, the Viterbi Algorithm for our Part of speech Tagger can be described as follows:

1. Set $P[1,s] = P(W1 = wi\ |t1 = ts)\ P(T1 = ts)P(T1 = ts) for\ 1 \le s \le k$
2. Set $L[1,s] = \{ts\} for\ 1 \le s \le k$
3. Set $P[r,s] = \max 1 \le j \le k\ \{P[r - 1, j]p(Wr = wi\ |Tr = ts)P(Tr = ts\ |Tr - 1 = tj)\ for\ 2 \le r \le n\ \ and\ 1 \le s \le k\ \}$
4. Set $L[r,s] = \ \{L[r-1,\ argmax\ 1 \le j \le k\ P[r-1,j]P(Wr = wi\ |Tr = ts)P(Tr = ts\ |Tr - 1 = tj)]\ , tl\ \}\ for\ 2 \le r \le n\ and\ 1 \le s \le k$
5. Then the most likely sequence of tags is given by $L[n, argmax\ 1 \le j \le k\ P[n,j]]$

It is easy to see that the running time of Viterbi Algorithm for Part of Speech Tagging is 0(nk2) which is much more efficient and consequently feasible

6 Implementation



In the specific case of our Part of Speech tagging, the tags are assumed to be the states and the words are assumed to be the outputs. Hence, our part of Speech taggers consists of:

I. $Ps(Ti)$: Probability of system starting in tag Ti
II. $Ps(Tj|Ti)$: Probability of the system transitioning from tag Ti to Tj.
III. $Pe(Wj|Ti)$: Probability of the system emitting output Wj in tag Ti.

Given a sequence of words, our Part of Speech tagger is interested in finding the most likely sequence of tags that generates that sequence of words. In order to accomplish this, our Part of Speech Tagger makes two simplifying assumptions:

- The probability of a words depends only on its tag. It is independent of other words and other tags,
- The probability of a tag depends only on its previous tags. It is independent of next tags and tags before previous tag.

Thus , given a sequence of n words W1W2…Wn , the most likely sequence of tags T1T2…Tn is

$$T_1T_2 \dots T_n =$$

$$\underset{T_1T_2 \dots T_n}{argmax} P(T_1T_2 \dots T_n | W_1W_2 \dots W_n) =$$

$$\underset{T_1T_2 \dots T_n}{argmax} P(W_1W_2 \dots W_n | T_1T_2 \dots T_n) \frac{P(T_1T_2 \dots T_n)}{P(W_1W_2 \dots W_n)} =$$

$$\underset{T_1T_2 \dots T_n}{argmax} P(W_1W_2 \dots W_n | T_1T_2 \dots T_n) P(T_1T_2 \dots T_n) =$$

$$\underset{T_1T_2 \dots T_n}{argmax} \prod_{i=1}^{n} P(W_i|T_i) \prod_{i=2}^{n} P(T_i|T_{i-1})\, P(T_1)$$

7 Experiments Results

The HMM model is trained with 28618 sentences from Brown Tag Corpus. The total number of tagged words available is 579513. For testing we have 2390 sentences and 36394 words whose tagging we decide using the built model. We already know the tags of test data set and we compare against model labeled tags to get the accuracy of model.

Training Data Set:

Number of Sentences	Number of words
28618	579513

Testing Data Set:

Number of Sentences	Number of words
2390	36394

After Model tagging

Total Tags	Correctly Tagged	Incorrectly Tagged	Accuracy
36394	35109	1285	96.4%

Sample Sentences which the model identifies correctly:

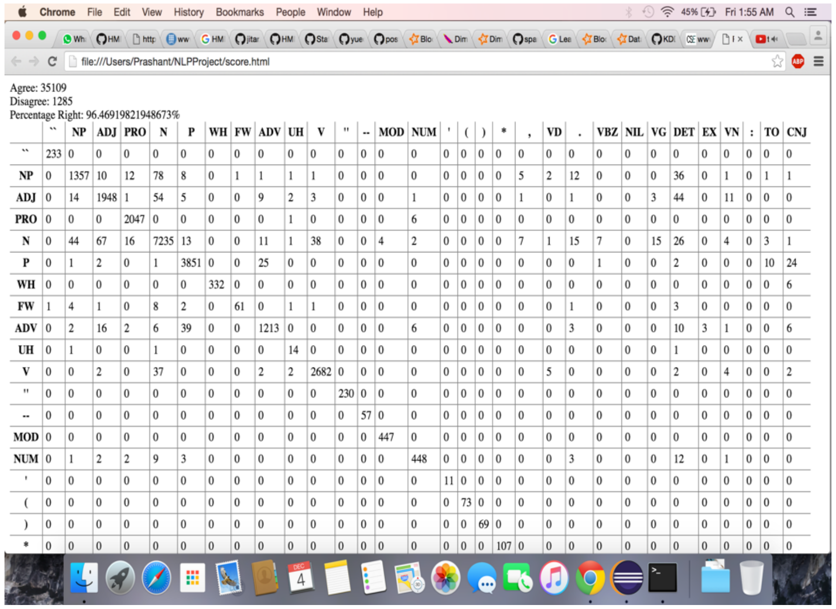
he said he had never talked to liston '' .

The tag given by model is : PRO VD PRO V ADV VN P NP '' .

It identifies liston as NP correctly because in the traning data we have sentence like :

liston is bill liston , baseball writer for the boston traveler , who quoted jensen as saying :

Also One scorer Class is written to understand which tags are getting wrong tagged mostly.


Chrome File Edit View History Bookmarks People Window Help
file:///Users/Prashant/NLPProject/score.html

Agree: 35109
Disagree: 1285
Percentage Right: 96.469198219486673%

	``	NP	ADJ	PRO	N	P	WH	FW	ADV	UH	V	''	--	MOD	NUM	'	(	)	*	,	VD	.	VBZ	NIL	VG	DET	EX	VN	:	TO	CNJ
``	233	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	0	0	0	0
NP	0	1357	10	12	78	8	0	1	1	1	1	0	0	0	0	0	0	0	0	5	2	12	0	0	0	36	0	1	0	1	1
ADJ	0	14	1948	1	54	5	0	0	9	2	3	0	0	0	1	0	0	0	0	1	0	1	0	0	3	44	0	11	0	0	0
PRO	0	0	0	2047	0	0	0	0	0	1	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	44	67	16	7235	13	0	0	11	1	38	0	0	4	2	0	0	0	0	7	1	15	7	0	15	26	0	4	0	3	1
P	0	1	2	0	1	3851	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	10	24
WH	0	0	0	0	0	0	332	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6
FW	1	4	1	0	8	2	0	61	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	3	0	0	0	0	0
ADV	0	2	16	2	6	39	0	0	1213	0	0	0	0	0	6	0	0	0	0	0	0	3	0	0	0	10	3	1	0	0	6
UH	0	1	0	0	1	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
V	0	0	2	0	37	0	0	0	2	2	2682	0	0	0	0	0	0	0	0	0	5	0	0	0	0	2	0	4	0	0	2
''	0	0	0	0	0	0	0	0	0	0	0	230	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	0	0	0	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MOD	0	0	0	0	0	0	0	0	0	0	0	0	0	447	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NUM	0	1	2	2	9	3	0	0	0	0	0	0	0	0	448	0	0	0	0	0	0	3	0	0	0	12	0	1	0	0	0
'	0	0	0	0	0	0	0	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	73	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	0	0	0	0	0	69	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	0	0	0	0	0	107	0	0	0	0	0	0	0	0	0	0	0	0

If we see the matrix given above, we found that Mostly Nouns get tagged as different part of speech and then adverb also gets tagged as different part of speech. System gets confused Nouns with Adjectives. The system is not that strong enough as its not trained with huge data set so if we provide some unknown word which is not in the training it is not able to predict the tag properly.

8 Conclusion

Various conclusion can be drawn after understanding the system and trying on different sentences.

1. The System performs really good with the test data provided.
2. All the miss tagged words, most of the words are noun tagged as adjectives and adverbs as pronouns. So for handling these scenarios the model should be strong.
3. Currently the training data set is less , if we train HMM model with more training data it means the system has seen more scenarios then the model will become strong and can handle more cases.
4. Currently the system behaves not that good with totally unseen words.
5. The system does not handle emotion tags.
6. For making system more scalable the model building should be done in distributed environment.

9 Future Work

Currently the system is very basic model using Viterbi Algorithm. We can enhance the system using three different kind of models.

1. Laplace Smoothed Hidden Markov Model
2. Absolute Discounting Hidden Markov model.
3. Interpolation Hidden Markov Model.

Using these different kinds of model will improve the system with unseen words and we can compare results with each other to see which performs better with different dataset.

References

Wikipedia page for POS tagging:
https://en.wikipedia.org/wiki/Part-of-speech_tagging

Wikipedia page for Hidden Markov Model:
https://en.wikipedia.org/wiki/Hidden_Markov_model

Part of Speech Explained by CMU :
https://www.cs.umd.edu/~nau/cmsc421/part-of-speech-tagging.pdf

Part of Speech Explained by UMASS:
https://people.cs.umass.edu/~mccallum/courses/inlp2004/lect10-tagginghmm1.pdf

Viterbi Algorithm Explanation:
http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/hmms.pdf

Natural Language Processing Final Project Report:
http://nlp.stanford.edu/courses/cs224n/2010/reports/parawira.pdf

Class Lecture by Dan Moldovan on POS Tagging from UTD:
http://www.hlt.utdallas.edu/~moldovan/CS6320.15F/Lecture5%20POS.pdf

Class Lecture by Dan Moldovan on Hidden Markov Model from UTD:
http://www.hlt.utdallas.edu/~moldovan/CS6320.15F/Lecture6%20HMM.pdf