PROJECT REPORT

Desaraju Harsha Vardhan 18084
Vijay Chakravarty 18309

Objectives:
To write a Python program to implement an HMM-based part of speech tagger on the English language.
To summarize an extensive error and success analysis of the tagger on known and unknown words.

Motivation: To acquire a thorough understanding of the Hidden Markov Model taught in class and to understand how dynamic programming algorithm of Viterbi search is helpful in the task of POS tagging.

Description:

HMM

An HMM allows us to talk about both observed events (like words that we see in the input) and hidden events or states (like part of speech tags) that we think of as causal factors in our probabilistic model.An HMM thus has two kinds of probabilities; the A transition probabilities of going from one state to the other and the B observation likelihoods of generating an observation given a particular state. There is a special start and end state which are not associated with observations.

Two simplifying assumptions in the HMM:
The probability of a word appearing is dependent only on its own part-of-speech tag; that it is independent of other words around it, and of the other tags around it.
The probability of a tag appearing is dependent only on the previous tag.It is the bigram assumption.

PROJECT REPORT

Viterbi algorithm

Suppose we are given a hidden Markov model (HMM) with state space S, initial probabilities π_i of being in state i and transition probabilities $a_{i,j}$ of transitioning from state i to state j. Say we observe outputs $y_1, \ldots, y_T$. The most likely state sequence $x_1, \ldots, x_T$ that produces the observations is given by the recurrence relations:[10]

$$\begin{aligned} V_{1,k} &= \mathrm{P}(y_1 \mid k) \cdot \pi_k \\ V_{t,k} &= \max_{x \in S} \left(\mathrm{P}(y_t \mid k) \cdot a_{x,k} \cdot V_{t-1,x} \right) \end{aligned}$$

Here $V_{t,k}$ is the probability of the most probable state sequence $\mathrm{P}(x_1, \ldots, x_t, y_1, \ldots, y_t)$ responsible for the first t observations that have k as its final state. The Viterbi path can be retrieved by saving back pointers that remember which state x was used in the second equation. Let $\mathrm{Ptr}(k, t)$ be the function that returns the value of x used to compute $V_{t,k}$ if $t > 1$, or k if $t = 1$. Then:

$$\begin{aligned} x_T &= \arg\max_{x \in S}(V_{T,x}) \\ x_{t-1} &= \mathrm{Ptr}(x_t, t) \end{aligned}$$

Dataset used:

Brown corpus

OBJ

Brown University Standard Corpus of Present-Day American English was compiled in the 1960 by Henry Kucera and W. Nelson Francis at Brown university.
It contains 500 samples of English-language text, totaling roughly one million words, compiled from works published in the United States in 1961.The Corpus consists of 500 samples, distributed across 15 genres in rough proportion to the amount published in 1961 in each of those genres. Each sample began at a random sentence-boundary in the article or other unit chosen, and continued up to the first sentence boundary after 2,000 words. In a very few cases miscounts led to samples being just under 2,000 words.

Experiments conducted:

Firstly, we divided the Brown corpus into a training dataset called “brown-train.txt” and a testing dataset called “brown-test.txt”. The code is run using these and the accuracy percentage is calculated for HMM_1.py and for HMM_2.py. These two files have two different strategies for dealing with unknown words.

PROJECT REPORT

After that, we found some ambiguous sentences or sentences that have difficult-to-tag words and manually tagged them in another file called “test.txt”. We replaced the “brown-test.txt” in the code with “test.txt” as the file to work with.
Hence, we analysed our ambiguous sentences separately and presented the analysis.

Findings and analysis:
We start our analysis with the sentence taken as an example in class for the HMM tagger. “Brown promises free.”

```
iiserb@cclab-OptiPlex-3040:~/Vij$ python3 HMM_1.py
[['.', '.'], ['free', 'jj'], ['promises', 'vbz'], ['Brown', 'np']]
UNKNOWN -  Transition from corpus

The accuracy is  100.0
```

Voila! That means we’re up to something. The HMM tagger is able to choose the sequence of tags that start with “brown” given the np tag instead of the more frequent jj tag because the entire sentence has greater probability when it starts with jj.

The Past Participle Problem (PPP)
Consider the following sentences.

1. The horse raced past the barn fell.
2. The whale named Willy was finally free.
3. The student awarded first place was overcome by joy.
4. Several competitors selected for the finals were unable to attend.

Past participles are tough to identify. That’s because in English, they usually look just like normal past tense verbs. Since normal verbs are much more common than participles, you aren’t expecting to see a participle when you start reading. When you see the first three words of the sentence—“the horse raced”—you naturally assume that you’ve just read the main subject and verb of the sentence. Then, when you find a second verb, you get confused. You have to go back and “re-understand” the first part of the sentence.

Like most of the native speakers of English, our HMM tagger too fails to correctly identify the past participle in the above sentences. The past participles “fell” and “named” are incorrectly marked as “vbd”, verbs in the past tense, instead of “vbn”,past

participiples. Interestingly though, the words “awarded” and “selected” are correctly tagged. We believe that this happened because in the corpus, the correctly tagged words appeared as past participles and therefore, are “known” words whereas the usage of the incorrectly tagged words as “vbn” doesn’t occur in the corpus.

```
iiserb@cclab-OptiPlex-3040:~/Vij/HSS222_project$ python3 HMM_1.py
[['.', '.'], ['fell', 'vbd'], ['barn', 'nn'], ['the', 'at'], ['past', 'in'], ['raced', 'vbd'], ['horse', 'nn'], ['The', 'at']]



[['.', '.'], ['free', 'jj'], ['finally', 'rb'], ['was', 'bedz'], ['Willy', 'np'], ['named', 'vbd'], ['whale', 'nn'], ['The', 'at']]



[['.', '.'], ['joy', 'nn'], ['by', 'in'], ['overcome', 'vbn'], ['was', 'bedz'], ['place', 'nn'], ['first', 'od'], ['awarded', 'vbn'],
 ['student', 'nn'], ['The', 'at']]



[['.', '.'], ['attend', 'vb'], ['to', 'to'], ['unable', 'jj'], ['were', 'bed'], ['finals', 'nns'], ['the', 'at'], ['for', 'in'], ['se
lected', 'vbn'], ['competitors', 'nns'], ['Several', 'ap']]



UNKNOWN -  Transition from corpus

The accuracy is  94.5945945945946
iiserb@cclab-OptiPlex-3040:~/Vij/HSS222_project$
```

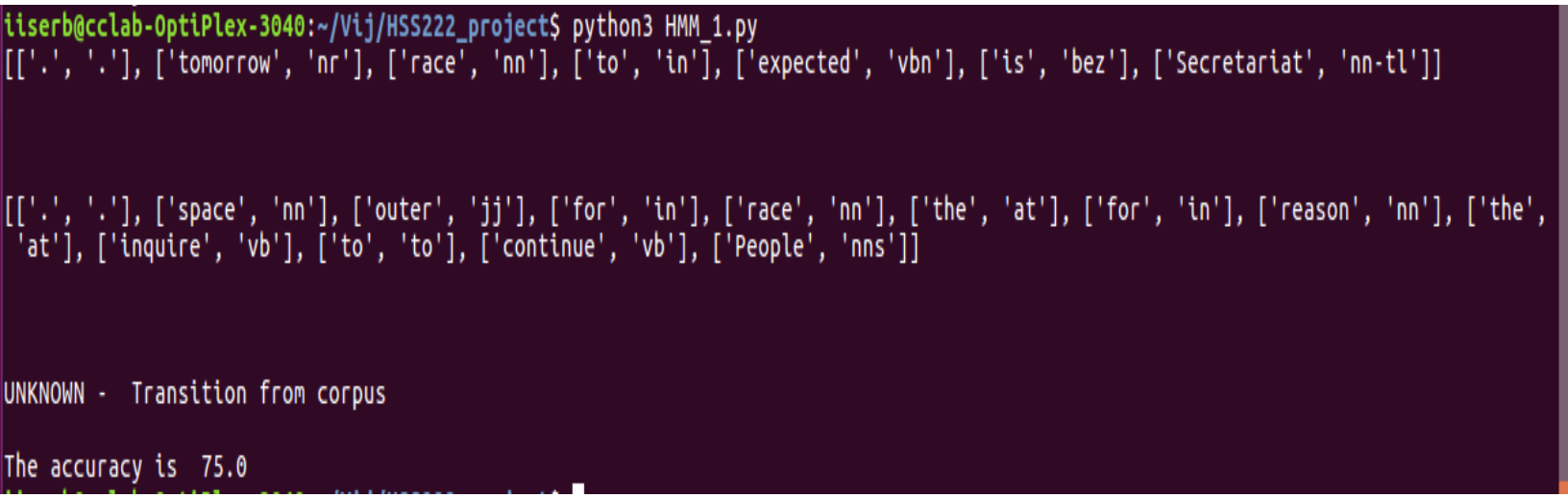
The Verb Noun Battle

Consider the following sentences.

1. Secretariat is expected to race tomorrow.
2. People continue to inquire the reason for the race for outer space.

Our HMM tagger does an impressive job of tagging the second sentence where the word “race” is used as a noun but it fails to recognise that the same word is used as a verb in the first sentence. The usage of “race” as a noun is predominant in the datasets available to us and therefore it’s a challenge for the tagger to learn whether to tag the word as a noun or a verb. The verb noun battle continues...

PROJECT REPORT



IN vs RB vs RP (the confusing trio)

1. He never got around to joining.
2. Spring is just around the corner.
3. Be a darling and show him around.
4. She has been a little off ever since the light went off.
5. The drag queen gave a stern look and off she went.
6. The settlement had vanished long since.
7. Since she studied, she topped.
8. Please turn around.
9. I turned my chair around to face the fire.

We look at the different usages of the word “around”. It gets correctly classified as a preposition in “around the corner” when it follows a noun and also correctly as “adverb” in “show him around”. But in the various sentences taken above, where “around” functions as a particle, we get the incorrect classification of “rb” (adverb).

Yet the word “since” gives us some hope in our tagger. It is correctly marked “in” (preposition) in “since the light went off” and as “cs” (subordinating conjunction) in “since she studied”.

The word “off” jumps easily over the hurdle “around” stumbled on. “Off” is classified as “rp” in “went off” and “off she went” but it fails to get it right as an adjective”jj” in “She was a little off” or “the fish was a bit off”. The tagger sees “off” in these sentences as “rp” (particle). It classifies them incorrectly as “rp” in other places as well where it is

```
[['.', '.'], ['joining', 'vbg'], ['to', 'in'], ['around', 'rb'], ['got', 'vbd'], ['never', 'rb'], ['He', 'pps']]


[['.', '.'], ['corner', 'nn'], ['the', 'at'], ['around', 'in'], ['just', 'rb'], ['is', 'bez'], ['Spring', 'np']]


[['.', '.'], ['around', 'rb'], ['him', 'ppo'], ['show', 'vb'], ['and', 'cc'], ['darling', 'nn'], ['a', 'at'], ['Be', 'be']]


[['off', 'rp'], ['went', 'vbd'], ['light', 'nn'], ['the', 'at'], ['since', 'in'], ['off', 'in'], ['little', 'jj'], ['a', 'at'], ['bee
n', 'ben'], ['has', 'hvz'], ['She', 'pps']]


[['!', '.'], ['went', 'vbd'], ['she', 'pps'], ['off', 'rp'], ['and', 'cc'], ['look', 'nn'], ['stern', 'jj'], ['a', 'at'], ['gave', 'v
bd'], ['queen', 'nn'], ['drag', 'nn'], ['The', 'at']]


[['.', '.'], ['since', 'rb'], ['long', 'rb'], ['vanished', 'vbn'], ['had', 'hvd'], ['settlement', 'nn'], ['The', 'at']]


[['.', '.'], ['topped', 'vbd'], ['she', 'pps'], [',', ','], ['studied', 'vbd'], ['she', 'pps'], ['Since', 'cs']]


[['.', '.'], ['around', 'rb'], ['turn', 'nn'], ['Please', 'vb']]


[['.', '.'], ['fire', 'nn'], ['the', 'at'], ['face', 'vb'], ['to', 'to'], ['around', 'rb'], ['chair', 'nn'], ['my', 'pp$'], ['turned'
, 'vbd'], ['I', 'ppss']]


UNKNOWN -  Transition from corpus

The accuracy is  89.04109589041096
jjserb@cclab-OptiPlex-3040:~/Vij$
```

functioning as a verb (“upped and offed”) or as a noun (“ready for the off”). We also notice a disturbing error. The words “please” and “supposedly” are incorrectly tagged as “vb” and “at” respectively.

```
[['.', '.'], ['firms', 'nns'], ['new', 'jj'], ['the', 'at'], ['to', 'in'], ['offed', 'nn'], ['and', 'cc'], ['upped', 'vbn'], ['sudden
ly', 'rb'], ['workers', 'nns'], ['loyal', 'jj'], ['Supposedly', 'at']]


[['.', '.'], ['off', 'rp'], ['the', 'at'], ['for', 'in'], ['ready', 'jj'], ['is', 'bez'], ['Ian', ','], ['Now', 'rb']]


[['.', '.'], ['days', 'nns'], ['off', 'in'], ['have', 'hv'], ['athletes', 'nns'], ['greatest', 'jjt'], ['the', 'at'], ['Even', 'rb']]


[['.', '.'], ['off', 'rp'], ['bit', 'nn'], ['a', 'at'], ['was', 'bedz'], ['fish', 'nn'], ['The', 'at']]
```

All the above observations of incorrect tagging remind us of the fact that our HMM tagger can only be as good as the dataset it is trained on. If a specific dataset,where words that can have multiple tags, is created, with adequate frequencies of usage of each of the different contexts a word might have, we would see better tagging ability of our tagger.

PROJECT REPORT

When a word is unknown and has a tag that is used rarely with that word, the problem gets compounded.It reminds us why it is important to create better and diverse corpora updated with the lingo used in contemporary times.

Unknown Words strategy

1. Did you hear about that buzzworthy bling of the Miss World?
2. The Youtuber is infamous for mansplaining.
3. And, as in uffish thought he stood, The Jabberwock, with eyes of flame,Came whiffling through the tulgey wood, And burbled as it came!

```
iiserb@cclab-OptiPlex-3040:~/Vij$ python3 HMM_2.py
[['?', '.'], ['World', 'nn-tl'], ['Miss', 'np'], ['the', 'at'], ['of', 'in'], ['bling', 'nn'], ['buzzworthy', 'nn'], ['that', 'dt'],
['about', 'in'], ['hear', 'vb'], ['you', 'ppss'], ['Did', 'dod']]


[['.', '.'], ['mansplaining', 'ppo'], ['for', 'in'], ['infamous', 'jj'], ['is', 'bez'], ['Youtuber', 'nn'], ['The', 'at']]


[['!', '.'], ['came', 'vbd'], ['it', 'pps'], ['as', 'cs'], ['burbled', 'vbd'], ['And', 'cc'], [',', ','], ['wood', 'nn'], ['tulgey',
'jj'], ['the', 'at'], ['through', 'in'], ['whiffling', 'rp'], ['Came', 'vbd'], [',', ','], ['flame', 'nn'], ['of', 'in'], ['eyes', 'n
ns'], ['with', 'in'], [',', ','], ['Jabberwock', 'nn'], ['The', 'at'], [',', ','], ['stood', 'vbd'], ['he', 'pps'], ['thought', 'nn']
, ['uffish', 'at'], ['in', 'in'], ['as', 'cs'], [',', ','], ['And', 'cc']]


UNKNOWN -  Transition as Emission

The accuracy is  89.79591836734694
```

The Brown corpus was compiled in the 1960s.Innumerable words have been added to the language since then. We used a couple of words from the modern-day millennial lingo (like “buzzworthy”, “bling” and “mansplaining”). We also used Jabberwocky speech done in class. “Bling”, “Youtuber”,”burbled”,”tulgey” and ”jabberwock” are correctly tagged. But “buzzworthy” has been marked “nn”, “whiffling” as “rp”,uffish as “at” and “mansplaining” as “ppo”. We believe that unknown words that are “nn” or “vbd” are more likely to get their correct tags due to their sheer high frequency.

We have tried two strategies to deal with unknown words. In the file “HMM_1.py”, we have used the unigram technique to calculate emission probabilities of unknown words. The probability of getting the unknown word given a tag, say “nn”, has been taken as the probability of occurrence of the tag “nn” in the training data. It has been assumed that all the unknown words are equally likely to have a given tag. The accuracy comes out to be 92.25%.

PROJECT REPORT

In the “HMM_2.py”, we have used the bigram technique to calculate emission probabilities of unknown words.The probability of getting an unknown word given a tag has been taken as the probability of getting that tag given the tag of the word preceding the unknown word. We see a slight improvement in the accuracy of the tagger using the bigram assumption. The accuracy comes out to be 92.37%.

Confusion Matrix:

```
iiserb@iiserb-OptiPlex-5050:~/Vij$ python3 HMM_1conf.py
UNKNOWN -  Transition from corpus
```

	nn	jj	pps	nns	vb	rb	rp	in	cc	cs	vbd	vbn	vbg	np
nn	95.67	0.99	1.62	0.00	0.05	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.16	0.16
jj	4.27	92.31	1.88	0.00	0.00	0.17	0.00	0.17	0.00	0.00	0.00	0.00	0.00	0.00
pps	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nns	5.60	0.14	1.40	91.18	0.00	0.00	0.00	0.28	0.14	0.00	0.00	0.14	0.00	0.00
vb	8.15	0.74	1.73	0.00	87.90	0.49	0.00	0.49	0.00	0.00	0.00	0.00	0.00	0.00
rb	0.89	2.22	2.22	0.00	0.00	88.00	0.00	2.22	0.44	0.89	0.00	0.00	0.00	0.00
rp	0.00	0.00	2.63	0.00	0.00	2.63	52.63	42.11	0.00	0.00	0.00	0.00	0.00	0.00
in	0.07	0.00	1.34	0.00	0.00	0.42	0.71	97.18	0.00	0.28	0.00	0.00	0.00	0.00
cc	0.00	0.00	0.70	0.00	0.00	0.00	0.00	0.00	99.30	0.00	0.00	0.00	0.00	0.00
cs	0.00	0.00	1.96	0.00	0.00	1.57	0.00	7.84	0.00	88.63	0.00	0.00	0.00	0.00
vbd	0.29	0.00	0.57	0.00	0.00	0.00	0.00	0.57	0.00	0.00	85.39	13.18	0.00	0.00
vbn	0.00	0.00	1.73	0.00	0.58	0.00	0.00	0.00	0.00	0.00	4.05	93.64	0.00	0.00
vbg	15.18	1.57	0.52	0.00	0.00	0.00	0.00	4.71	0.00	0.00	0.00	0.00	78.01	0.00
np	7.77	1.64	3.14	0.00	0.00	0.00	0.00	0.30	1.05	0.00	0.00	0.00	0.00	82.96

```
The accuracy is  92.25614863321404
```

For the HMM_1.py file, we get the above confusion matrix.
When we look at the leading diagonal of the matrix, we find numbers nearing 90% and above. This gives the accuracy percentage of each tag. The no. in the ith row and the j th column gives us the percentage of inaccurately tagging the tag in the ith row as the tag in the jth column. Only “pps” is tagged as “pps” all the time. The next best performer is “cc” with 99.3% accuracy, the rest 0.7% being incorrectly tagged as “pps”.”in” has a performance of 97.18%,”nn” has a performance of 95.67%. We see a trend-the most frequent tags are more likely to be tagged correctly. Our worst performer is “rp” with just 52.63% accuracy. Particles are difficult even for native speakers.Therefore, it should not come as a great surprise that 42.11% of the times an “rp” is

being incorrectly tagged as “in” and the rest of the times as “rb” or “pps”. This is the same problem that we discussed in “RP vs RB vs IN”.

For the HMM_2.py file, we get the following confusion matrix.

UNKNOWN - Transition as Emission

	nn	jj	pps	nns	vb	rb	rp	in	cc	cs	vbd	vbn	vbg	np
nn	94.95	1.25	1.62	0.00	0.10	0.00	0.05	0.05	0.05	0.00	0.05	0.00	0.16	0.26
jj	3.08	92.99	1.88	0.00	0.00	0.17	0.00	0.17	0.00	0.00	0.00	0.17	0.00	0.00
pps	0.00	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nns	3.22	0.14	1.40	91.60	0.00	0.28	0.00	0.28	0.14	0.00	0.14	0.14	0.00	0.00
vb	7.90	0.74	1.73	0.00	88.15	0.49	0.00	0.25	0.00	0.00	0.25	0.00	0.00	0.00
rb	0.89	2.22	2.22	0.00	0.00	88.00	0.00	2.22	0.44	0.89	0.00	0.00	0.00	0.00
rp	0.00	0.00	2.63	0.00	0.00	2.63	52.63	42.11	0.00	0.00	0.00	0.00	0.00	0.00
in	0.07	0.00	1.34	0.00	0.00	0.42	0.71	97.11	0.00	0.28	0.00	0.00	0.00	0.07
cc	0.00	0.00	0.70	0.00	0.00	0.00	0.00	0.00	99.30	0.00	0.00	0.00	0.00	0.00
cs	0.00	0.00	1.96	0.00	0.00	1.57	0.00	7.84	0.00	88.63	0.00	0.00	0.00	0.00
vbd	0.00	0.00	0.57	0.00	0.00	0.00	0.00	0.29	0.00	0.00	85.67	12.89	0.00	0.00
vbn	0.00	0.00	1.73	0.00	0.58	0.00	0.00	0.00	0.00	0.00	4.05	93.64	0.00	0.00
vbg	13.61	1.57	0.52	0.00	0.00	0.52	0.00	1.05	0.00	0.00	0.00	0.52	79.58	0.00
np	1.64	2.69	4.04	0.00	0.00	0.15	0.00	0.00	0.30	0.00	2.09	0.00	0.00	84.30

The accuracy is 92.3779791365263 %

Tags like “jj”, “nns”, “vb”, “in”, “vbd”,”vbg” and “np” perform better with the bigram assumption for unknown words while the performance of “nn” goes down. “Pps”,”rb”,”rp”,”cc” and “cs” and “vbn” perform just like they did with unigram assumption for unknown words.
The introduction of bigram consideration increases the confusion of “nn” with “jj”. When we have two consecutive words like “glass container”,both our taggers classify both “glass” and “container” as “nn”. But when we replace the word “glass” with an unknown word “carborundum”, the tagger with unigram assumption still correctly classifies both “carborundum” and “container” as “nn”. However, the tagger with bigram assumption for unknown words falls for the bait and classifies “carborundum” as “jj” instead of “nn” because it is more likely for an adjective to precede a noun than a noun to precede a noun. The beautiful observation has been shown below.

PROJECT REPORT

```
iiserb@iiserb-OptiPlex-5050:~/Vij$ python3 HMM_1.py
[['.', '.'], ['fell', 'vbd'], ['container', 'nn'], ['glass', 'nn'], ['The', 'at']]


UNKNOWN -  Transition from corpus

The accuracy is  100.0
iiserb@iiserb-OptiPlex-5050:~/Vij$ python3 HMM_2.py
[['.', '.'], ['fell', 'vbd'], ['container', 'nn'], ['glass', 'nn'], ['The', 'at']]


UNKNOWN -  Transition as Emission

The accuracy is  100.0
iiserb@iiserb-OptiPlex-5050:~/Vij$ python3 HMM_2.py
[['.', '.'], ['fell', 'vbd'], ['container', 'nn'], ['carborundum', 'jj'], ['The', 'at']]


UNKNOWN -  Transition as Emission

The accuracy is  80.0
iiserb@iiserb-OptiPlex-5050:~/Vij$ python3 HMM_1.py
[['.', '.'], ['fell', 'vbd'], ['container', 'nn'], ['carborundum', 'nn'], ['The', 'at']]


UNKNOWN -  Transition from corpus

The accuracy is  100.0
```

Implications of our findings and possibilities for future work:

Our HMM tagger (92% accuracy) has been pretty successful as compared to the unigram(89% accuracy) or bigram tagger(82% accuracy) for the 500 sentences we tagged from the Brown corpus. This shows us that the HMM tagger provides a better way to tag sentences and deal with unknown words. However, we find that the HMM tagger too has the following drawbacks:
Multiple associations between tags and words can’t be managed.
It is inflexible to memorize multiple spans of sentences.
Inaccuracies in tagging some ambiguous words still persists and that gives us the motivation to come up with better modelling and algorithm techniques to carry out the task of POS tagging.We have used hand tagged data in our tagger. We could have also used EM algorithm to train an HMM tagger on unlabelled data. For future work, we can explore giving more context to our tagger by using trigram probabilities. To deal with the problem of data sparsity, techniques like deleted interpolation can be used. Another approach could be to use Transformation based tagging to derive insights from both stochastic and rule based taggers, thereby exploiting the best of both Worlds.Furthermore, a slightly more complex algorithm for dealing with unknown words could be based on the idea that the probability distribution of tags over unknown words is very similar to the

distribution of tags over words that occurred only once in a training set. One could also use the morphology of unknown words to get cues to tag them.