HW02 Project Report

1. Penn Tree Bank Tagging

The below sentences are tagged with Penn Treebank Parts of Speech Tags.

1.1 Exercise 5.1

1. I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NNP
2. Does/VBZ this/DT flight/NN serve/VBP Dinner/NNS
3. I/PRP have/VBP a/DT friend/NN living/VBG in/IN Denver/NNP
4. Can/MD you/PRP list/VB the/DT nonstop/JJ afternoon/NN flights/NNS

1.2 Exercise 5.2

1. It/PRP is/VBZ a/DT nice/JJ night/NN ./.
2. This/DT crap/NN game/NN is/VBZ over/IN a/DT garage/NN in/IN Fifty-second/NNP Street/NNP ...
3. ... NoBody/NN ever/RB takes/VBZ the/DT newspapers/NNS she/PRP sells/VBZ ...
4. He/PRP is/VBZ a/DT tall/JJ ,/, skinny/JJ guy/NN with/IN a/DT long/JJ ,/, sad/JJ ,/, mean-looking/JJ kisser/NN ,/, and/CC a/DT mournful/JJ voice/NN ./.
5. ... I/PRP am/VBP sitting/VBG in/IN Mindy/NNP 's/POS restaurant/NN putting/VBG on/RP the/DT gefillte/NN fish/NN ,/, which/WDT is/VBZ a/DT dish/NN I/PRP am/VBP very/RB fond/JJ off/RP ,/, ...
6. When/WRB a/DT guy/NN and/CC a/DT doll/NN get/VBP to/TO taking/VBG peeks/NNS back/RB and/CC forth/RB at/IN each/DT other/JJ ,/, why/WRB there/EX you/PRP are/VBP indeed/RB ./.

3. Maximum Entropy Tagging

In maximum entropy models, the sequence tagging is done as a multi-label classification task either by doing forward or backward traversal of the given input sequence.

In Ratnaparkhi's MaxEnt tagger, the forward procedure for tagging the sequence does not always find the best possible sequence of tags because

1. The features for each word in the seq. is the context of 2 words to the right and 2 words to the left and tags of previous 2 words. This means it fails to consider the long distance relationships (outside the context) which might have an influence.
2. As the model depends on previous 2 tags (t_{i-1}, t_{i-2}) if any of them is tagged wrongly then the model might tag the words that comes next incorrectly.

At the same time, backward procedure also does not yield perfect result always. Moreover, it is difficult to combine the information from forward and backward procedure to tag the word. In order to overcome this issues, we need a model that maximizes the likelihood of joint tagging for a given input sequence. Probabilistic Sequence models like Conditional Random Fields, Hidden Markov Model with Viterbi algorithm can be used to collectively determine the most global assignment of tags for the given words in the sequence. Neural Network based approaches like RNN would also help for sequence processing tasks.

5. POS Tag Entropy

The brown corpus is a collection of 500 American English documents spread across various categories whereas Treebank corpus has sentences from Wall Street Journals. The words in these corpora are tagged with parts of speech. The tagged corpora is preprocessed (say X_{pre}) by converting the words into lower case and there by ignoring the words that appear less than 5 times. The tag entropies for all the words in the corpora are computed.

5.1 Top 10 Highest Entropy Words

The words are sorted based upon their tags entropies and the top 10 words are reported below with the format as Word Count_of_the_word Tags Entropy

5.1.1 Brown Corpus

northeast 16 {('northeast', 'NR-TL-HL'), ('northeast', 'NN-TL'), ('northeast', 'NR-TL'), ('northeast', 'NR'), ('northeast', 'JJ'), ('northeast', 'JJ-TL')} 2.452819531114783
chase 18 {('chase', 'NN-TL'), ('chase', 'NN-HL'), ('chase', 'VB'), ('chase', 'NP'), ('chase', 'NP-TL'), ('chase', 'NN')} 2.3718956694089632
round 75 {('round', 'NN-TL'), ('round', 'NN'), ('round', 'RB'), ('round', 'VB'), ('round', 'IN'), ('round', 'JJ'), ('round', 'JJ-TL'), ('round', 'JJ-HL')} 2.345722433710929
b 105 {('b', 'NN-HL'), ('b', 'NN-TL-HL'), ('b', 'NP-HL'), ('b', 'NP'), ('b', 'NP-TL'), ('b', 'NN'), ('b', 'NN-TL')} 2.2787839998066235
right 613 {('right', 'JJ'), ('right', 'NN-HL'), ('right', 'NR'), ('right', 'QL'), ('right', 'RB'), ('right', 'NN'), ('right', 'NN-TL'), ('right', 'JJ-HL'), ('right', 'JJ-TL')} 2.245358265403839
northwest 25 {('northwest', 'NR'), ('northwest', 'NN-TL'), ('northwest', 'JJ-TL'), ('northwest', 'NR-TL'), ('northwest', 'JJ')} 2.197405811425518
pro 16 {('pro', 'NN'), ('pro', 'FW-IN-TL'), ('pro', 'IN-HL'), ('pro', 'JJ'), ('pro', 'IN')} 2.149397470347699
beat 68 {('beat', 'NNS'), ('beat', 'NN-TL-HL'), ('beat', 'VB'), ('beat', 'VBD'), ('beat', 'VBN'), ('beat', 'JJ-TL'), ('beat', 'JJ'), ('beat', 'NN')} 2.0998469626866143
gross 66 {('gross', 'NN'), ('gross', 'JJ-HL'), ('gross', 'JJ'), ('gross', 'JJ-TL'), ('gross', 'NP-TL'), ('gross', 'NP')} 2.0573358641495325
rival 12 {('rival', 'NN'), ('rival', 'NN-HL'), ('rival', 'VB'), ('rival', 'JJ'), ('rival', 'JJ-TL')} 2.054585169337799

5.1.2 Penn Treebank

hit 10 {('hit', 'VBP'), ('hit', 'VBD'), ('hit', 'VBN'), ('hit', 'NN'), ('hit', 'VB')} 2.170950594454669
set 29 {('set', 'VBP'), ('set', 'VBD'), ('set', 'VBN'), ('set', 'NN'), ('set', 'VB')} 2.0719814254906335
close 28 {('close', 'NN'), ('close', 'JJ'), ('close', 'VB'), ('close', 'RB')} 1.8409745639875459
savings 24 {('savings', 'NNPS'), ('savings', 'NNS'), ('savings', 'NN'), ('savings', 'NNP')} 1.8337993233858494
lead 20 {('lead', 'VB'), ('lead', 'JJ'), ('lead', 'NN'), ('lead', 'VBP')} 1.765957320949175
limited 10 {('limited', 'VBN'), ('limited', 'JJ'), ('limited', 'NNP'), ('limited', 'VBD')} 1.7609640474436812
down 57 {('down', 'NN'), ('down', 'IN'), ('down', 'RP'), ('down', 'NNP'), ('down', 'RB')} 1.7605902374234237
run 18 {('run', 'VB'), ('run', 'VBN'), ('run', 'NN'), ('run', 'VBP')} 1.7527152789797045
call 8 {('call', 'VB'), ('call', 'VBP'), ('call', 'NN'), ('call', 'NNP')} 1.75
cut 28 {('cut', 'VBD'), ('cut', 'VBN'), ('cut', 'NN'), ('cut', 'VB')} 1.7449047091342316

5.2 Discussion

Entropy is a measure of uncertainty. As the tag entropy of a word is high, the more uncertain it is in getting the right tag. So, the words that have a wide range of possible tags will have high entropy and is difficult to tag right always. The same trend has been observed in both the corpora as shown above.

Apart from this, for each corpus the lists of words based upon the no of different tags that the word takes are computed. The counts for the words that take more than 1 tag are shown in Figure 1 and for Penn Treebank and Brown corpus. It's observed that, the total no of different words having ambiguous tags are 7224 and 1005 for brown and tree bank respectively. Out of them, 68.3% and 77.2% of the words are can possibly have 2 tags for brown and Penn Treebank respectively. So, its easy to disambiguate the tags for these words because the various tags associated with a word are not equally likely. Also, as the number of possible tags that a word takes increases, the overall count of those words decreases.

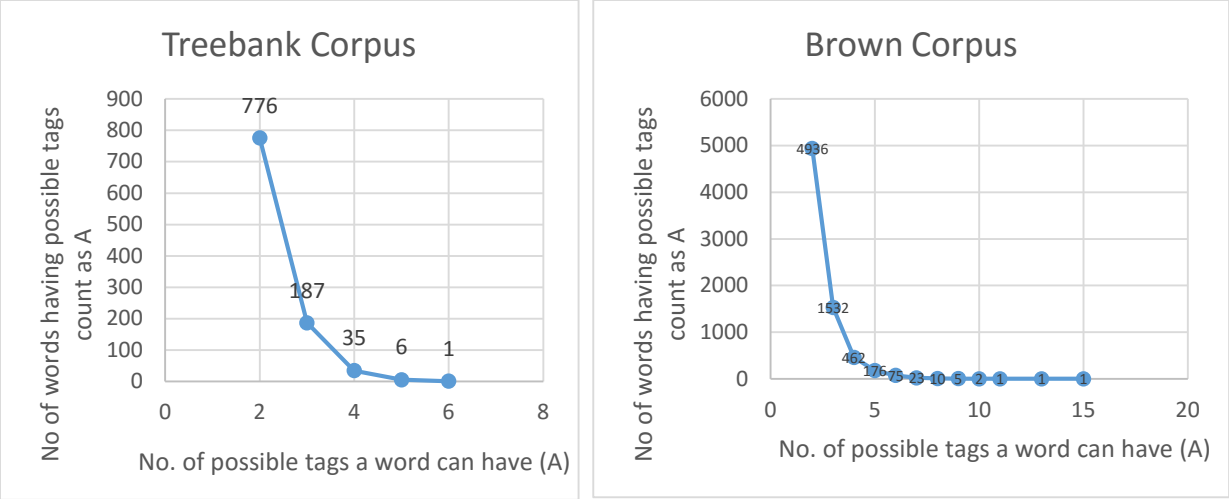


Figure 1 Penn Treebank and Brown Corpus6. HMM POS Tagging

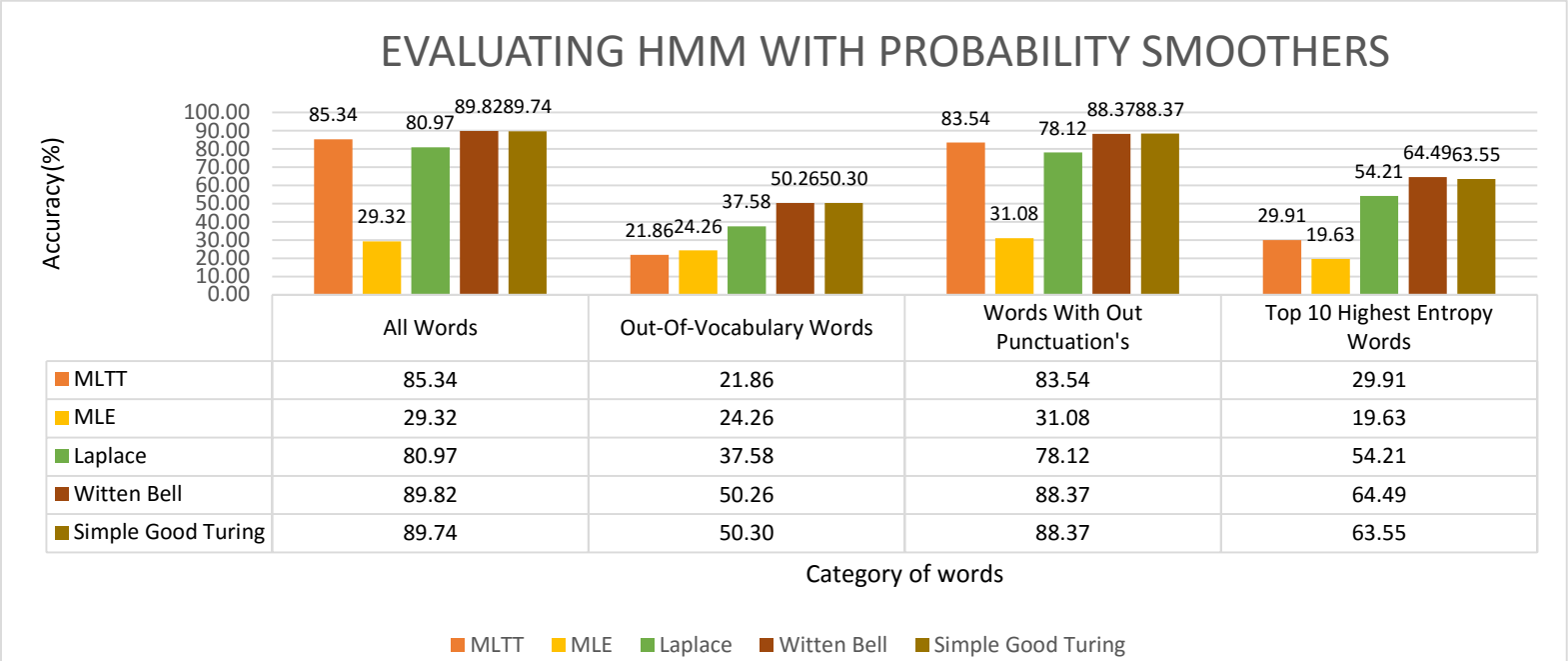
6. HMM POS Tagging

Treebank corpus has a collection of text files from Wall Street Journal. All the sentences in these text files are tagged with Penn Tree Bank POS tags. No preprocessing of the data is done for the below experiments. Tag Entropies are computed for all the words in the corpus and the top 10 highest entropy words are *{set, cut, forecast, close, put, hit, down, try, range, Plains}*. In this Exercise, I have tested and compared Hidden Markov Model (HMM) with different probability estimators and a base line system Most Likely Tag Tagger on 4 different categories of words in test data. I have compared accuracies of these 4 categories of words using Viterbi Tagging and Most Likely Tag using $\gamma_j(t)$ for best HMM model. I have also experimented to determine the effect of the size of training corpus for learning an HMM.

6.1 Evaluating multiple models for HMM POS

The HMM is tested for parts of speech tagging with MLE, Laplace, Witten Bell and Simple Good Turing probability estimators. Along it with it, a base line system Most Likely Tag Tagger (MLTT) is used to compare with above mentioned HMM's. In MLTT for each word, a tag is assigned by picking the tag that is most observed with that word in the training data and for out-of-vocabulary words "NN" will be assigned as tag. All the HMM's here use Viterbi Algorithm. The accuracies of these 5 different models are reported in Figure 2 when the following categories of words 1. All words 2. Out-of-vocabulary words 3. Words without punctuation's 4. Top 10 entropy words are considered. Figure 3 clearly shows that smoothing has a key role to play in HMM training. When all words in the test data are considered, there is an absolute increase of 50% from MLE to Laplace. Similar kind of trend has been observed in case #2, #3 and #4.

The base line system MLTT has reported very good accuracies when compared to HMM with MLE estimates. This could be because when all words or words without punctuations in the testing data are used most of the words will possibly have a single tag or 2 which helps MLTT to give the right tag always. For out-of-vocabulary (oovs) words as MLTT always give 'NN' which is could not be the right tag for most of the oovs. So, the accuracy is less when compared to MLE. For Top 10 Highest entropy words, we would have a wide range of possible tags for each word. So, it is tough for this base line system to assign the right tag and reported an accuracy of 29.91% but still it's better than MLE which reported just 19.63%.



For this data set, MLTT is even better than HMM with Laplace Smoothing when all words or words without punctuations are considered

We know that Simple Good Turing and Witten Bell Smoothing do better probability estimates when compared with Laplace which is basic idea of smoothing. When all the words in the test data are considered, the best accuracy 89.82% is reported by HMM with Witten Bell Smoothing followed by Simple Good Turing 89.74%. When words without punctuations are considered, Witten Bell and Simple Good Turing reported the same accuracy (88.37%). For out-of-vocabulary words, there is an increase in accuracy from HMM with Laplace to HMM with Simple Good Turing estimates. The reason is that smoothing mainly targets sparsity and HMM gets better smoothing when moving from Laplace to Simple Good Turing estimates and this explains the better performance reported in case of out-of-vocabulary words.

6.2 Most Likely Tag using $\gamma_j(t)$ Approach

In HMM, the probability $\gamma_j(t)$ of being in state j at given time t and μ is computed as $\alpha_j(t)*\beta_j(t)/P(O/\mu)$ where $\alpha_j(t)$ and $\beta_j(t)$ are the forward and backward probabilities respectively and $P(O/\mu)$ is the probability of having an observation sequence O. In case of parts of speech tagging, the states of an HMM are tags whereas sequence of observations over T time steps are sequence of words. So, this approach instead of back tracing like Viterbi, it assigns the words with tags that are most likely based upon the $\gamma_j(t)$ values.

6.2.1 Results and Discussion

The results are reported in Figure 3 and shows that Viterbi out performs Most Likely Tag on all the 4 categories of words of the test data and this is because Viterbi always choose the best path while assigning tags to the sequence of words. In case of Most Likely Tag approach, while assigning the tag to a

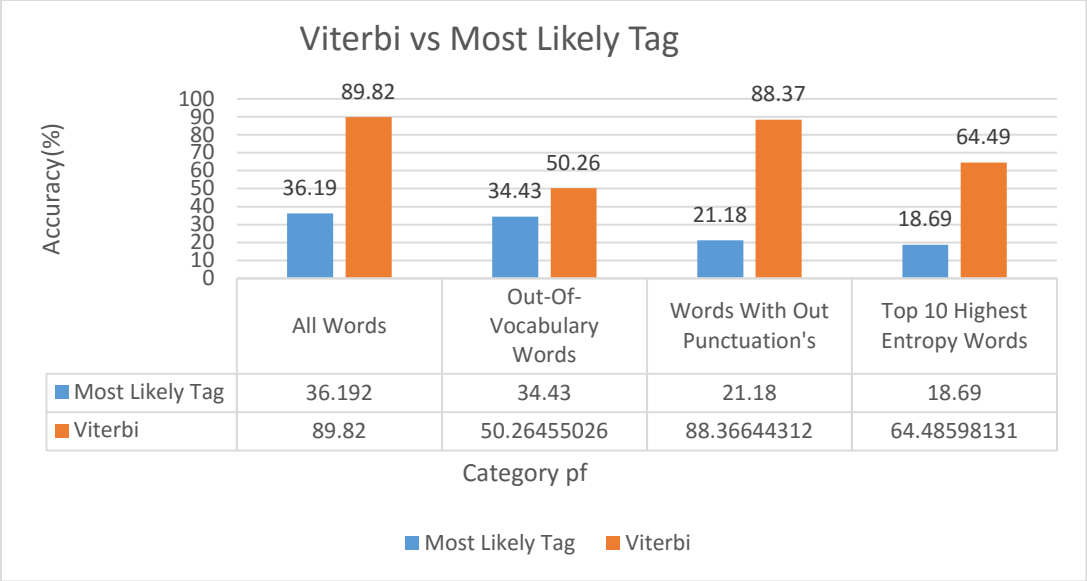


Figure 3 Viterbi vs Most Likely Tag

word in a sequence, it just takes the most probable tag over all the possible tags and this might not yield the correct result.

6.3 Learning Curve

The best HMM model is when Witten Bill smoothing estimates are considered. The learning curve is plotted in Figure 4 by using 10% to 100% of the files in training data in the increments of 10 with 100% test data on Penn Treebank. It is observed that as the size of the training data increases, the performance on test data increases. There are many parameters like transition probabilities, symbol emission probabilities and start state probabilities that the model has to learn from training dataset. In order to have good estimates of these values, the model needs to have good amount of training data. So, we can see there is a jump in accuracy from 68% to 77.73% when there is an increase of just 10% of the training data.

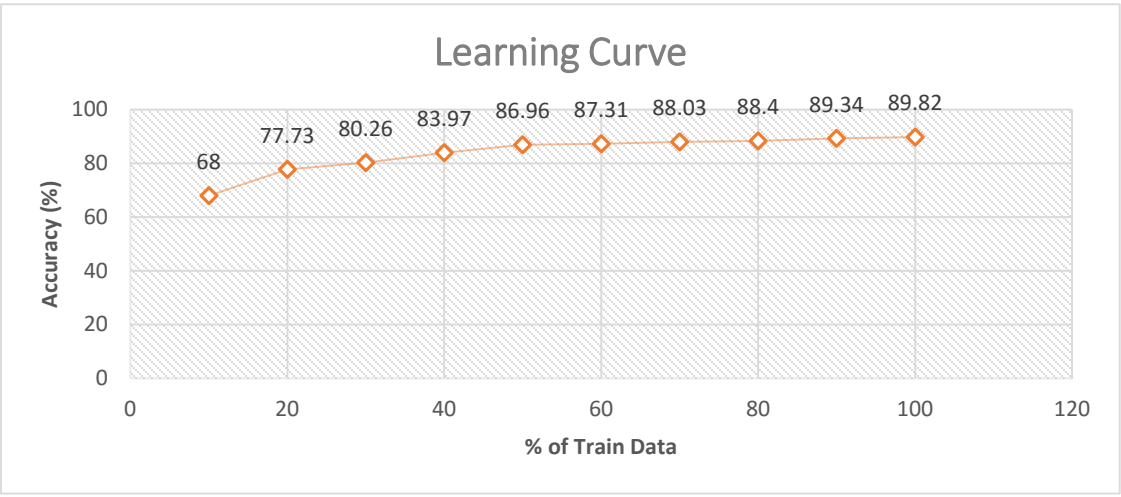


Figure 4 Learning Curve