Toy Problem 1.1

	START	G	C	A	C	T	G	END
H	0	0.18	0.0216	0.001728	0.00020736	0.0000165888	0.0000023328	0.0000011664
L		0.08	0.0144	0.002592	0.0002592	0.00003888	0.000003888	

The blue arrows are the back pointers in the trellis table

	START	G	C	A	C	T	G	END
H	0	0.18	0.0216	0.001728	0.00020736	0.0000165888	0.0000023328	0.0000011664
L		0.08	0.0144	0.002592	0.0002592	0.00003888	0.000003888	

Final Hidden state of the given DNA sequence- {H,H,L,L,L,L}

Viterbi Variable	Backpoiter
V1(H)=0.18	B1(H)=START
V1(L)=0.08	B1(L)=START
V2(H)=0.0216	B2(H)=H
V2(L)=0.0144	B2(L)=H
V3(H)=0.001728	B3(H)=H
V3(L)=0.002592	B3(L)=H
V4(H)=0.00020736	B4(H)=H
V4(L)=0.0002592	B4(L)=L
V5(H)=0.0000165888	B5(H)=H
V5(L)=0.00003888	B5(L)=L
V6(H)=0.0000023328	B6(H)=L
V6(L)=0.000003888	B6(L)=L
V7(END)=0.0000011664	B7(END)=L

POS Tagging 1.2

1.2.1

Value of K=6

Size of Final Vocab including special unknown token UNK = 10143

1.2.5

Viterbi Algorithm

I have created a function for Viterbi algorithm where I pass as parameter each sentence of the dataset separating the tags each time alongwith the smoothened Emission Probability and the Transmission Probabilty. For the first token calculated the Viterbi variable by the sum of Emission Probability and Transmission Probability for each tag. Then calculated the Viterbi variable for each token by calculating the max of (Vm-1+S(Ym,Ym-1)) where Y is the tag and m is the current token and hence m-1 the previous token. For the best Viterbi variable stored the respective back pointer as this would help us get the sequence of tag for each sentence. At the end we backtracked the backpointer to retrieve the sequence.

Accuracy on Dev data (with alpha=1 and beta=1)= approx. 87%

Conditional Random Fields 2

Features Added –

1. First Letter of the token
2. Last Letter of the token
3. First two letter of the token
4. Last two letter of the token
5. Previous word of the token
6. Next word of the token
7. First three letter of the token
8. Last three letter of the token

9. Is the token a number
10. Is the token a special character

File	Accuracy on Train	Accuracy on Dev
6 new features with lbgfs	0.697638	0.694303
More than 6 features with lbgfs	0.750695	0.747113
With Averaged Perceptron	0.972098	0.874789

2.2-> As logistic regression is a probabilistic model, it takes the class with the highest probability of being true. For perceptron model, we train the model on the weight of each feature . When the model wrongly predicts an instance to a class we update the weight and predict by dot product of the weights and the features.

For using Average Perceptron to train CRF, we have Viterbi score as one of the feature and add features like we did previously like 1st and last letter of the token etc. and have weights for each feature and have the same characteristic of perceptron of giving penalty when the model predicts a tag wrong by updating its weights. Each word that would be run in the Average Perceptron would be a feature and the feature set would include features like First Letter of the token, Last Letter of the token, First two letter of the token, Last two letter of the token, Previous word of the token and emission and transmission probability that we calculate for calculating the Viterbi score.