201 hw3

Darian Lee

December 2024

1 part 1

QUESTION 1: Why does $\arg\max_t P(t \mid w) = \arg\max_t \log P(t, w)$?

Anwser: Because log is a monotonic function so the maximums and minimums will be the same as in the original function.

QUESTION 2: For every value of t_j, let $\pi_j(t_j)$ be the log probability of the highest scoring tag sequence of length j ending in tag t_j:

$$\pi_j(t_j) = \max_{t_1,\ldots,t_{j-1}} \sum_{i=1}^{j} \text{score}(w, i, t_i, t_{i-1})$$

Show that $\pi_j(t_j)$ can be computed using π_{j-1}. Answer:

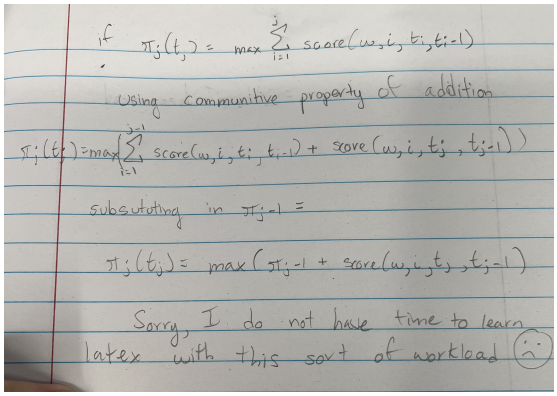

Figure 1: q2

QUESTION 3: Give an algorithm for computing $\hat{t}$ defined in the last line of Eq. (1). What is the time complexity of the algorithm? Be sure to carefully argue your answer.

QUESTION 4: Show that $\pi_j(t_j)$ can again be computed using π_{j-1}. Which semiring properties did you use in your proof? Describe the changes needed to your algorithm in question 2 to solve Eq. (2).

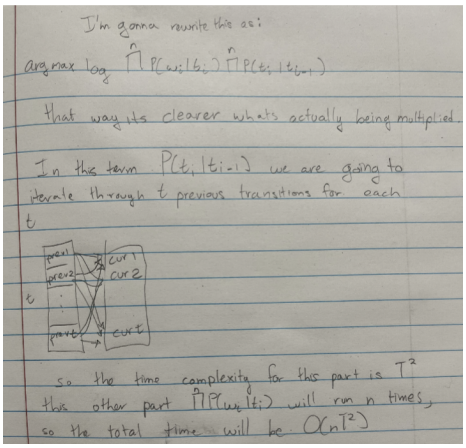

Figure 2: q3

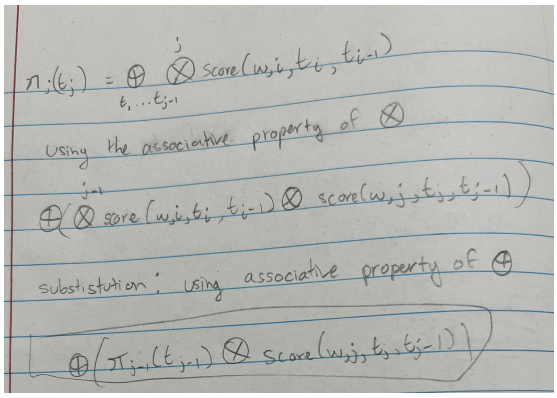

Figure 3: q4

QUESTION: Programming Part Testing my model on the test set using the transition and emission probabilities calculated from the training set was relatively slow. However, the final accuracy and F1 scores were very high, indicating a strong model. This model was compared to a baseline that only used emission probabilities and a greedy search. Interestingly, the greedy search outperformed the Viterbi algorithm in terms of F1, precision, and recall, while the Viterbi algorithm performed better in terms of overall accuracy. This could be because the greedy algorithm, by selecting the most probable tag for each token independently, may focus more on making correct decisions for individual tokens, resulting in higher precision, recall, and F1 scores.

The baseline model performed much worse on all metrics, which is expected given its reliance solely on emission probabilities, causing it to underfit the data.

The algorithm could be improved by combining Viterbi and greedy search

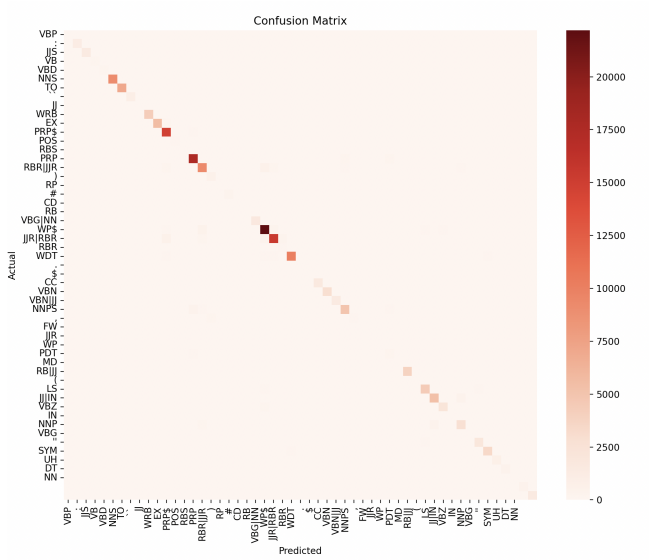

Figure 4: Confusion matrix for greedy search

(e.g., using Viterbi on some sentences and greedy search on others) to improve time complexity and F1 scores. The high results from the greedy search suggest that Viterbi might not be necessary on this dataset to achieve good results. A better approach might involve using Viterbi only for sentences where there is significant uncertainty, with a threshold for how similar the probabilities need to be in the greedy search before switching to Viterbi.

The most commonly confused pairs were NN (singular noun) and JJ (adjective), VBD (verb, past tense) and VBN (verb, past participle), and NNP (proper noun, singular) and NNPS (proper noun, plural). NN and JJ were likely confused because both often follow a determiner. VBD and VBN are similarly confusing because they both describe verbs in the past tense, while NNP and NNPS tend to appear in the same contexts as well.

Tag 1	**Tag 2**	**Count**
NN	JJ	420.0
VBD	VBN	398.0
NNP	NNPS	318.0
NN	NN	305.0
JJ	NN	305.0

Table 1: Most Commonly Confused Tag Pairs in Viterbi Model

QUESTION: Experimenting with Different Alphas When testing different values of alpha on my dev set, I decided to use only a 100-sample subset of the dev data due to the slow execution of my code. After testing alpha values

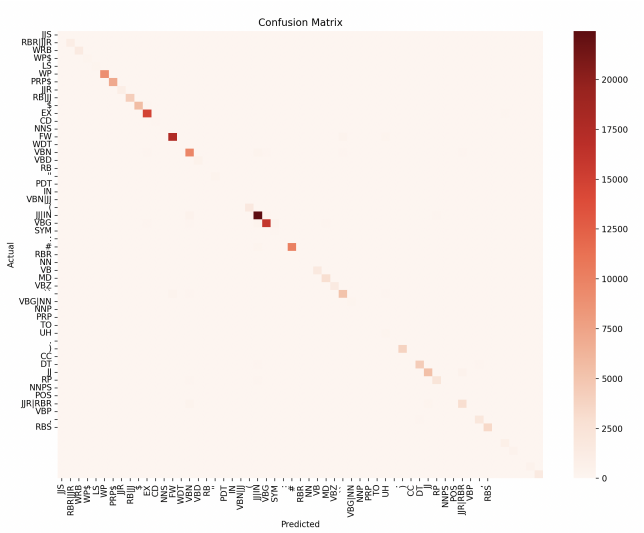

Figure 5: Confusion matrix for viterbi

Model	Precision	Recall	F1-Score	Accuracy
Baseline (Emission Probabilities Only)	0.5159	0.5825	0.5234	0.8831
Greedy	0.7526	0.7260	0.7279	0.9402
Viterbi	**0.7111**	**0.7154**	**0.7104**	**0.9506**

Table 2: Model Evaluation Results

between 1 and 5, I found that the best alpha was 1, yielding an F1 score of 0.9217 and an accuracy of 0.9475. In general, as the value of alpha increased, the evaluation metrics decreased. This could be because higher alpha values resulted in excessively high probabilities for unseen transition and emission pairs, leading to overfitting.

Alpha	**Precision**	**Recall**	**F1 Score**	**Accuracy**
1	0.9213	0.9267	0.9217	0.9475
2	0.9206	0.9265	0.9213	0.9470
3	0.9150	0.9084	0.9064	0.9449
4	0.9152	0.9098	0.9072	0.9454
5	0.9145	0.9098	0.9069	0.9454

Table 3: Performance Metrics for Different Alpha Values

Since I achieved the same optimal alpha value on my test set as I did initially on the dev set, there was no need to rerun the tests.

To improve the testing speed, one possible approach is to use a greedy method or to implement beam search to decrease the space being searched.