1 Algorithm

1.1

Defining a forward variable $\alpha_m(y_m)$,

$$\alpha_m(y_m) = \sum_{y_1:m-1} \exp \sum_{n-1}^{m} s_n(y_n, y_{n-1})$$

$$\alpha_m(y_m) = \sum_{y_1:m-1} \prod_{n-1}^{m} \exp s_n(y_n, y_{n-1})$$

Similar to max-product algorithm in Viterbi's algorithm, we can use sum-product algorithm.

$$\alpha_m(y_m) = \sum_{y_1:m-1} \prod_{n-1}^{m} \exp s_n(y_n, y_{n-1})$$

$$\alpha_m(y_m) = \sum_{y_1:m-1} (\exp s_m(y_m, y_{m-1})) \prod_{n-1}^{m} \exp s_n(y_n, y_{n-1})$$

$$\alpha_m(y_m) = \sum_{y_1:m-1} (\exp s_m(y_m, y_{m-1})) * \alpha_{m-1}(y_m - 1).......(i)$$

In following algorithm K is length of tags and M is length of the sentence. We initialise dp array with scoring function of start tag as follow,

$$dp(y_1) = s_1(y_1, start)$$

where start is a start of sentence.

Algorithm 1: Each $s_m(k; k_0)$ is a local score for tag $y_m = k$ and $y_{m-1} = k_0$

```
for k ∈ 0, ...., K do
    dp(k) = s_1(k, start) end
    for m ∈ 2, ...., M do
        for k ∈ 0, ....K do
            dp_m(k) = sum exp s_m(k, k') + dp_{m-1}(k')
        end
    end
```

1.2 Justification

Because recurrence requires computing a sum over K terms at each node in our DP solution; i.e. we generate k values for each node by adding k values from previous nodes. Therefore, the time complexity id $O(TK^2)$,
Where, T is length of sentence and K is length of possible tags.

2

2.1 Convert the estimated probabilities from the previous step into log space and implement the Viterbi decoding algorithm as in Algorithm 11in [Eisenstein, 2018]. Report the accuracy of your decoder on the dev set dev.pos

Best accuracy on dev set is 0.9566347008709313

2.2 Tune and to find the best accuracy on the dev set. Report the best accuracy number and the values of and .

Best value for and are 1 and 1 respectively; giving accuracy 0.95 on both dev and tst set.

2.3 Run the model selected from the previous step on the test set tst.pos, and report the accuracy

Best accuracy on test set is 0.9590364050774317