2 Most frequent tag baseline

2.b

Our F1 Score on the dev set is 0.80.

3 HMM tagger

3.b

- The optimal are $\lambda_1 = 0.12, \lambda_1 = 0.6$.
- Our pruning policy includes two parts: pruning when emission is zero and pruning when transition is zero. The key to both strategies is that we don't need to pay attention to paths with zero probability if a part of a path has zero probability. We also use caching and a different kind of iterating on tags. More concretely, we skip iterating on tags that have zero emission probability and for the transition, for each y_i which is v from the pseudo code we iterate only on the y_{i-2}, y_{i-1} which are w, u from the pseudo code that have a non zero probability of being selected last round of the process. I.e. their $\pi[k-1]$ value is bigger than 0 ($-\infty$ in log space). This happens only if they where documented in $\pi[k-1]$. We will show this claim using induction. Our claim is that for each possible path of length k, we have its possible last two elements documented in $\pi[k-1]$. We start by having all the possible part-paths of length 0 documented in $\pi[-1]$ by having $\pi[-1](*,*) = 1$. Assuming correctness for $k-1$ we will show that it is true for k. We iterate over all tags (y_i/v), we skip if the emission is zero. But then it not a part of a path. If the emission is not zero, we iterate on all possible values y_{i-2}, y_{i-1} to be the last two elements in a path before it. If the transition probability is zero so it is not part of a path (markovian assumption). Otherwise, we put y_{i-1}, y_i in $\pi[k]$. If there was any other possible path that has some y_{i-1}, y_i then by our induction hypothesis, there were y_{i-2}, y_{i-1} that we would iterate over and get a positive emission and transition, thus putting them in $\pi[k]$.

3.c

Our F1 score on the dev set is 0.84.

3.d

We will take the sentence "A-a B-b" and assume we only have a, b as our tags. So if $e(A|a) = 0.5, q(a|*,*) = 0.4, e(A|b) = 0.5, q(b|*,*) = 0.6, q(b|a,*) = 1$ etc. we would first choose $A \longrightarrow b$ rather than $A \longrightarrow a$ as we should. Thus, the greedy algorithm is not correct.

4 Maximum Entropy Markov Model (MEMM) tagger

4.d

We have used the same methods that we have used in question 3. This time, we also used pruning for partial paths with probability that is less that -2 in log space and we calculated the vector transformation and logreg for all of the prev tag and prev-prev tag in a single shot.

4.e

The F1 scores we got on the development set for the greedy algorithm is 0.89. We got the same F1 for the viterbi algorithm but the viterbi is a bit better. It got the same recall and 0.01 points higher in precision.

4.f

YOUR ANSWER HERE

5 BiLSTM tagger

5.b.i

YOUR ANSWER HERE