**NLP – Ex3**

**Q2**

b. Our accuracy on the dev set is 92.6%

**Q3**

c. accuracy on dev set is 95.55

**Q4**

c. We incorporated a few optimizations in our implementation:

- When extracting features for position i, we first extracted features with “mock” tags of ‘#’ (for extracting the features related to the input words like prev\_word, prefixes etc.). Then, for each prev\_tag and prevprev\_tag we just updated the features dictionary with the correct tags and saved if separately.

- We predicted scores for all t, u, v options in specific position i with one call to logreg.predict\_log\_proba(). It seems that it does its own optimizations such that predicting score for one example and for multiple examples together takes pretty much the same time. Then, when calculating the score for each t, u, v triplet we took the proper score from the predicted scores.

- for each k, u, v we took the maximum score (and back pointer) in-place, without going over all possibilities of t after all evaluations.

d. Our accuracy on the dev set is 95.89% with the Viterbi algorithm and 95.87% with the greedy algorithm.

e. We sampled errors from the MEMM model. We noticed some failure cases:

* Tagging rare words replaced with a category. For example:
  + “…,and **lowerCase** from it…”: True tag: VBN, model tag: NN
  + ” …Shearson's **'UNK'**, UNK…”: True tag: JJ, model tag: NN
  + “…“Where We **initCap**” commercials…”: True tag: VBP, model tag: NNPS

A reasonable explanation to this can be that the categories are too general so there is a lot of variance in the true tags in the data.

* Confusion between VBN and JJ. Sometimes it seems that the true label was the one that is incorrect:
  + “Fidelity, for example, **prepared** ads several…” True tag: VBN, model tag: JJ
  + “…,when **frightened** investors flooded…”: True tag: JJ, model tag: VBN
  + “'The **complicated** language in…”: True tag: VBN, model tag: JJ (here it seems that actually the model was right, and the ground truth is wrong)