CSE 556, Natural Language Processing
Monsoon, 2019
Assignment 3
Deadline: 23:59 hrs, 29th September, 2019

Instructions:

- **Please don't copy from internet or any other student. Plagiarism will thoroughly be checked on all the submitted files. Refer: Academic Dishonesty Policy**
- **If any python code file is found to be in text/pdf file format, assignment will not be graded or the graded assignment will be allotted 0.**
- **Allowed Programming language: Python**
- **Use classroom discussion for any doubt.**
- **Total points for this assignment is 100.**
- **Please find the dataset for the assignment here: Link**

Task 1: Hidden Markov Model [50 points]

Implement an HMM-based approach to POS tagging. Specifically, you have to implement the Viterbi algorithm using a bigram tag/state model. For training, a POS-tagged section of the BERP corpus has been attached(**Training set_HMM.txt**). Your systems will be evaluated against an unseen test set drawn from the same corpus.

Training

The training data consists of around 15,000 POS-tagged sentences from the BERP corpus. The sentences are arranged as one word-tag pair per line with a blank line between sentences, words and tags are tab-separated. Contractions are split out into separate tokens with separate tags. An example is shown here:

I	PRP
'D	MD
Like	VB
french	JJ
Food	NN

Assume that the tags that appear in the training data constitute all the tags that exist (no new tags will appear in testing). On the other hand, new words may appear in the test set.

Decoding

For decoding, your system will read in sentences from a file with the same format minus the Tags, ie, one word per line ending with a period and a blank line before the next sentence. As output you should emit an appropriate tag for each word in the same format as the training data.

Task 2: Maximum Entropy Modelling [50 points]

Implement multinomial logistic regression for noun group tagging.

Training

Use the **train.np** file to train your model. There are a total of around 10,000 unique words and 50 unique tags in the corpus. Assume that the tags that appear in the training data constitute all the tags that exist (no new tags will appear in testing). On the other hand, new words may appear in the test set. Each line constitutes the word, POS-tag and Noun group tag. The noun group tags are among: B-NP, I-NP and O

- B-NP – indicates a token begins a Noun Group
- I-NP – indicates a token is inside a Noun Group
- O – indicates a token is outside of a Noun Group

You must use the following features for the purpose of training your model:

1. Start of sentence.
2. End of sentence.
3. Previous POS tag (Consider all the POS tags that is associated with the word past the current word in the entire corpus)
4. Next POS tag (Consider the POS tag that is associated with the word following the current word with highest frequency in the entire corpus)
5. BIO-tag in the training set.

The value of weights at every stage should be the value of probability:

Count satisfying the feature / Total such count

Eg.

1. Count of I at start of sentence / Total count of I in the corpus.
2. Count of previous tag as NN / Count of all NN tags in the corpus.

Decoding:

Use **dev.np** file to test your model. The dev.np file contain word, POS-tag and Noun group tag(BIO-tag). As output you should emit an appropriate tag for each word in the same format as the training data, ie, you should use the word and predict the Noun group tag for the word using your model and create a new file **output.np**, which will have word, POS-tag, BIO-tag and your predicted BIO-tag [POS and BIO-tags can directly be read from the dev.np file]. (Have a look at **sample.test** for an example)

Report your accuracy for each of the BIO-tags, which should be calculated as follows:
$\text{Accuracy}_{\text{B-NP}}$ = # correctly classified B-NP tags / # B-NP tag in dev.np

Similarly for I-NP and O tags.

Note:

1. Train you model and save it in pickle file. Your model should not be trained during demo.
2. Handle OOV words in some systematic way.

What to Submit?

A zip/tar file containing:

- **Running code.**
- **Trained model**
- **Report. Should have:**
  - **Preprocessing steps used.**
  - **State all you assumptions.**
  - **Observations, if any.**
  - **Accuracy values**