CSE 842: NATURAL LANGUAGE PROCESSING
HOMEWORK 2 ASSIGNMENT

Due Date: October 18, 2024 at 11:59 PM

PROBLEM 1: HIDDEN MARKOV MODEL (HMM)

Description: For this problem, you are to calculate the transition & emission probability matrices for an HMM. You do *not* have to code the full HMM or Viterbi decoding, just the calculations to get these two matrices.

Code Requirements:

1. Calculate the MLE for the transition probabilities (A) matrix of an HMM. Remember these are the probabilities that we would move from one state (POS tag) to another.
2. Calculate the MLE for the emission probabilities (B) matrix of an HMM. Remember these counts represent how often we see a specific word with a tag.
3. Dataset: use the *news* category of the Brown corpus, with the *universal* tagset. This tagset has 12 classes: Noun, Verb, Pronoun, Adjective, Adverb, Adpositions, Conjunctions, Determiner, Cardinal Numbers, Particles, Other, Punctuation.
   ```
   brown.tagged_words(categories='news', tagset='universal')
   ```

What to submit:

1. Your Python files or Jupyter notebook implementing the calculation of transition (A) matrix & emission (B) matrix. Please name your file as *yournetid_hmm.py.*
2. In the written report, include how to run your code and a sample of the 2 matrices.
   a. For the A matrix, report the entire table (12 tags in rows & 12 tags in columns with transition probabilities in the cells).
   b. For the B matrix, report a subset of the calculations. Your table should have 12 rows, one for each POS tag. The columns should represent the emission probabilities for the following words: science, all, well, like, but, and 2 meaningful words of your choice (e.g., your name, home city/country, favorite color, etc.). Note: some words might not appear in the corpus subset though.

PROBLEM 2: NEURAL NETWORK FOR PART-OF-SPEECH TAGGING

Description: For this problem you will need to implement and evaluate a neural network, specifically an RNN or LSTM, for POS tagging. Use any deep learning framework of your choice. I recommend trying Keras with scikit-learn. (Other frameworks are faster & allow more experimentation, but Keras is fast to understand.)

Tutorials: The following tutorials may be helpful in getting started.
TensorFlow & Keras: RNN
Keras: Step by step
Keras: Multi-class classification
PyTorch: Classifying Names with a Character-Level RNN

Code Requirements:

1. Neural Net: Implement an RNN or LSTM using a deep learning framework of your choice, sklearn, numpy, nltk, etc.
2. Cross-validation: Use an 80%-20% training-testing split for cross-validation.
   Optional (not required): use an 80-10-10 training-development-test split.
3. Features to test: capitalization, word position in sentence (first or last), word contains numbers & letters, word has a hyphen, entire word is capitalized, word is a number, first four characters of suffixes & prefixes (e.g., -ed, -ing, -ous), Glove embeddings, etc. You must incorporate at least 5 of these features. Using more will improve model performance.
4. Hyperparameters to test: number of hidden layers, optimizers (e.g., Adam, Adagrad, SGD, …), dropout rates, activation functions (sigmoid vs. ReLu vs. tanh), embeddings (dimension size, pre-trained vs. learned), learning rates, etc. You must test at least 3 different hyperparameters.
5. Dataset: Use the NLTK Treebank with the *universal* tagset.
   `tagged_sentences = nltk.corpus.treebank.tagged_sents(tagset='universal')`

General implementation tips: Depending on your modeling framework, you might need to initialize the hidden layer with either zeroes or nonzero values. Features will need to be represented as vectors and DictVectorizer might be useful (depends on your processing). Sentences have different lengths, but you'll need to use a fixed length. There are different ways to handle this including: take a maximum length from the dataset, or take the average and make it a little longer. Once you choose a length, you need to do padding if the input sentence is too short, or some form of shortening if it's too long.

What to submit:

1. Your Python code or notebook implementing your *best* NN model, including feature representation, training, testing, and evaluation; name the file *yournetid_nn.py*.
2. In the written report, include: your best model's architecture, how to run your code, features tested, hyperparameters tested. Using a table format, report the *accuracy* of your NN under the different features & hyperparameters.

FOR THE LAST 10 POINTS

Again, this is your choice. You could try implementing all the features for Problem 2, using your HMM to do PoS tagging, implementing a Bi-LSTM, using a non-English or different dataset for tagging, etc. Choose something you're interested in trying out that's relevant to the lectures/assignment.

SUBMISSION SUMMARY

Please submit the Python files or notebook(s) that solve problems 1 & 2 and *one* written report in **PDF** format named *yournetid_hw2_report.pdf*. If you zip your files together, please include your NetID in the filename.