MIT 6.806/6.864: Advanced Natural Language Processing **Fall 2021**

Homework 3

Due Date: Friday, October 22 11:59 PM EST (*Canvas submission*)

HMMs and Trees

Introduction. In this assignment, you'll work with two models of linguistic structure: *finite-state* models (hidden Markov Models, or HMMs) and trees. After training each model, you'll evaluate its implicit assumptions and answer theoretical questions about how some sequences are modelled successfully. Finally, you will detail your answers and experimental results in a report, which will be submitted on **Friday, October 22, 11:59 PM**. Your report should also include analysis as to the significance and reasoning behind your experimental results.

We provide scaffolding code for each part of the lab in two Jupyter notebooks; the code for Part 1 is in https://colab.research.google.com/github/mit-6864/hw3/blob/main/6864_hw3_hmms_student_version.ipynb and the code for Part 2 is in https://colab.research.google.com/github/mit-6864/hw3/blob/main/6864_hw3_trees_student_version.ipynb. We recommend saving a copy of each notebook and using Google Colab to write and execute your code.

General report guidelines. Homework assignments should be submitted in the form of a research report on Canvas. Please upload a single PDF of your report concatenated with print outs of your code (e.g., the Jupyter Notebooks with your implementation). The report section of your submitted PDF should consist of a maximum of four single-spaced pages (6.806) or six single-spaced pages (6.864) typeset with LaTeX.[1] Reports should have one section for each part of the assignment below. Each section should describe the details of your code implementation and include whatever analysis and figures are necessary to answer the corresponding set of questions.

Part 1: Hidden Markov Models

In Part 1, you'll use the Baum–Welch algorithm to learn *categorical* representations of words in your vocabulary. It should briefly discuss any implementation details that were important to filling out the code part of the homework.

(a) (*Computational*) Consider the following corpus, with vocabulary size 4, which was used in one of the provided test cases:

$[0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2]$
$[0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3]$

[1] If you'd like, you can use the Association for Computational Linguistics style files, available at: https://2021.aclweb.org/downloads/acl-ijcnlp2021-templates.zip

$[1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2]$
$[1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3]$

Construct a HMM so that these four sequences are the only possible sequences of length 12 that could be generated, by providing the values of A, B, and π associated with the model.

(b) (*Experimental*) What do the learned hidden states seem to encode when you run unsupervised HMM training with only 2 states? What about 10? What about 100?

(c) (*Experimental*) As in HW 2, what's the relationship between the number of labeled examples and usefulness of HMM-based sentence representations? Why or why not might HMM state distributions be sensible sentence representations?

(d) (**6.864 students only**; *Theoretical*) Consider the classic n-gram model for $n = 2$ (a bigram model), and suppose we train this model on a corpus with vocabulary size v. Show that you can implement this bigram model as a v-state HMM.

Part 2: Trees

(a) *(Theoretical)* In this question, you will be tasked with writing a *Context Free Grammar* (CFG) for a subset of English language that includes simple intransitive sentences (1) and those with an arbitrary amount of object relative ("that") clauses (2-4). (Note: the "that" is optional in normal English but required for this problem.)

Example Sentences:

(1) The man ran

(2) The man [that the cat chased] ran

(3) The man [that the dog [that the cat feared] chased] ran

(4) The man [that the dog [that the cat [that the man loved] feared] chased] ran

Our Vocabulary:

- Nouns = {*man,dog,cat*}
- Intransitive verbs = {*meowed, barked, ran*}
- Transitive verbs = {*feared, chased, loved*}
- Determiner = {*the*}
- Conjunction = {*that*}

(i) For the first part of the problem assume that there can be an infinite number of embedded object-relative clauses in a sentence. Note that the top level sentence always has an intransitive verb. Please provide the CFG rules that can generate this

subset of the English language (please ignore the brackets). Here is a rule to get you started: S $\rightarrow$ NP VP .

(ii) Now, let's consider the case where only up to double nesting of object-relative clauses is allowed (i.e. (3) is fine but (4) isn't). How could you modify your grammar to obtain this language? Please provide the CFG rules.

Part 2 of your lab report should briefly discuss any implementation details that were important to filling out the code part of the homework. Then use the code to answer the following questions:

Part A

(a) (*Experimental*) What are your final scores? (Please provide `F1`, `exact_match`, and `tree_match` results). What percent of the predictions are well formed?

(b) (*Experimental*) We generate the word and span embeddings with a bi-directional LSTM. Why is this better than using a uni-directional LSTM?

(c) (*Experimental*) Can you suggest one potential method to improve the current algorithm for generating span embeddings? Explain why it is possible to do better.

Part B (6.864 only)

(a) (**6.864 students only**; *Experimental*) What are your final scores? Did you see any improvement over the baseline model in Part A? What's the reason for the drop in F1 scores? (Please provide `F1` and `exact_match`, `tree_match` results.)

(b) (**6.864 students only**; *Experimental*) In results, find some trees that have the correct tree structure but incorrect labeling. Then, print the labeling mistakes for those examples. Qualitatively comment on the mistakes you observe.

(c) (**6.864 students only**; *Experimental*) In training, we label some random spans without labels as `None`. How does that help with our decoding?

(d) (**6.864 students only**; *Theoretical*) Is it easy to frame the whole problem as a seq2seq task? How would you do it?