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

Homework 1

Due Date: Thurs. March 11, 2021, 12:00 PM (noon) EST (*Canvas submission*)

Word Representations

Introduction. In this assignment, you'll explore three different ways of using unlabeled text data to learn pretrained word representations. For each of the three representation learning schemes, you'll implement, train, and evaluate the scheme on a real dataset, then prove and discuss some theoretical properties. Finally, you'll submit a report describing both experiments and answers to the theoretical and computational questions in this handout. Your report should also describe the effects of different modeling decisions (representation learning objective, context size, etc.) on both qualitative properties of learned representations and their effect on a downstream prediction problem. We provide scaffolding code for each part of the lab in two Jupyter notebooks; the code for Parts 1 and 2 are in `6864_hw1_part_1-2.ipynb` and the code for Part 3 is in `6864_hw1_part_3.ipynb` (these notebooks are available in the following GitHub repository: `https://github.com/mit-6864/hw1`). We recommend saving a copy of each notebook and using Google Colab to write and execute your code.[1]

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.[2] 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: Matrix Factorization

The first part of your lab report should discuss any important details and design decisions you made when implementing the LSA featurizer. Additionally, you should design and conduct experiments to answer the following questions in your report:

(a) (*Theoretical, Computational*) Recall that the we can compute the word co-occurrence matrix $W_{tt} = W_{td}W_{td}^\top$. What can you prove about the relationship between the left sin-

[1] You can open the notebook for Parts 1 and 2 in Google Colab with this link: `http://colab.research.google.com/github/mit-6864/hw1/blob/main/6864_hw1_part_1-2.ipynb`; the notebook for Part B is available at: `http://colab.research.google.com/github/mit-6864/hw1/blob/main/6864_hw1_part_3.ipynb`.

[2] 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`

gular vectors of W_{td} and W_{tt}? Do you observe this behavior with your implementation of `learn_reps_lsa`? Why or why not?

(b) (*Experimental*) Qualitatively, what do you observe about nearest neighbors in representation space? For example, what words are most similar to *the*, *dog*, *3*, and *good*? How does the size of the LSA representation affect this behavior?

(c) (**6.864 students only**; *Experimental*) Do learned representations help with the review classification problem? What is the relationship between the number of labeled examples and the effect of word embeddings?

Part 2: Language Modeling

Part 2 of your lab report should discuss any implementation details that were important to filling out the code above. Then, use the code to set up experiments that answer the following questions:

(a) (*Experimental*) Qualitatively, what do you observe about nearest neighbors in representation space? (E.g. what words are most similar to *the*, *dog*, *3*, and *good*?) How well do Word2Vec representations correspond to your intuitions about word similarity?

(b) (*Experimental*) How do results on the downstream classification problem compare to Part 1?

(c) (**6.864 students only**; *Experimental*) One important parameter in Word2Vec-style models is context size. How does changing the context size affect the kinds of representations that are learned?

Part 3: Hidden Markov Models

In the remaining part of this assignment you'll use the Baum–Welch algorithm to learn *categorical* representations of words in your vocabulary. Answers to questions in this lab should go in the same report as the initial release.

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

$[0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2]$
$[0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3]$
$[1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2]$
$[1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3]$

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 before, what's the relationship between the number of labeled examples and usefulness of HMM-based sentence representations? Are these results generally better or worse than in Parts 1 and 2 of the homework? 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.