Assignment 2

CS 421: Natural Language Processing

Due: September 25, 2020 (12 p.m. CST)

1. Introduction

In this assignment, you will learn about Hidden Markov Models, n-grams, and Naive Bayes classification.

Hidden Markov Models (often abbreviated as *HMMs*) are statistical models used to represent Markov processes with unobservable (or hidden) states. HMMs are guided by the simple *Markov property* that the future is dependent only on the present, rather than on all of the history leading up to it. Common applications of these models in NLP include POS tagging and automatic speech recognition (ASR). HMMs also have a wide range of applications in machine learning, bioinformatics, and other engineering disciplines. These models are explained in great detail in Appendix A of the course textbook.

A statistical **language model** (LM) defines a probability distribution over sequences of symbols (words or characters). Given a sequence of symbols of length t, it can assign a probability to that sequence $P(w_1, w_2, ..., w_t)$. Statistical language models are widely used in NLP and related fields and are a fundamental building block in most systems. Some popular applications for language models include machine translation, auto-complete, authorship attribution, and optical character recognition. An **n-gram language model** is a language model that computes these probabilities using fixed length sequences of words, or *n-grams*. Up until a few years ago, n-gram based language modeling was the dominant approach. However, recently neural probabilistic language models have become extremely popular. You can read more about n-gram language models in Chapter 3 of the course textbook, and we will study neural language models in a few weeks.

Text classification is a fundamental task in NLP. It involves assigning a label from a predefined set of labels to an input text. For instance, you may think of product reviews as input text, where each product review can be assigned a binary label indicating whether the review is positive or negative. Text classification is an example of a *supervised machine learning* task, in which the objective is to learn a statistical model (given labeled training data) that has the capacity to map new input data to its correct label. **Naive Bayes** is a basic yet scalable and efficient probabilistic classifier which utilizes Bayes' theorem for (in our case) text classification; however, it can be applied to other

domains as well. The "naive" in Naive Bayes indicates the strong assumption that features are independent. This is not necessarily true—as we've seen with language models, words (or features) are dependent on the context. However, if we "naively" assume that is not the case, then we can use Naive Bayes for text classification. Despite its strong simplifying assumption, Naive Bayes exhibits reasonable performance for simple text classification tasks. You can read more about Naive Bayes in Chapter 4 of the course textbook.

Please follow the instructions in Section 2 to solve all the questions listed in Section 3.

2. Instructions

Each question is labeled as `Code` or `Written` and the guidelines for each type are provided below.

Code

The `Code` questions need to be completed using Python (version 3.6+). There are no **external** packages required to complete this assignment. If you want to use an external package for any reason, you are required to get approval from the course staff on Piazza prior to submission. Templates are provided for each `Code` question (.py files) as supplementary material. Do not rename/delete any functions or global variables provided in these templates and write your solution in the specified sections. Use the `main` function (provided in templates) to test your code when running it from a terminal. Avoid writing that test code in the global scope, however, you should write additional functions/classes as needed in the global scope. These templates may also contain important information and/or examples in comments so please read them carefully. This part of the assignment will be graded automatically using Gradescope.

To submit your solution for `Code` questions, you need to compress the following files (after completion) in a single `zip` file. These files should be in the root of your `zip` archive for autograding to work correctly.

- ☐ `q1.py`
- ☐ `q2.py`

Submit this `zip` file on Gradescope under `Assignment 2 - Code`. All specified files need to be submitted to receive full credit.

Written

You are required to submit all `Written` questions in a single PDF file. You may create this PDF using Microsoft Word, scans of your handwritten solution, LaTeX or any other method you prefer.

To submit your solution for `Written` questions, you need to provide answers to the following questions in a single PDF.

☐ Q2(b)

☐ Q3

Before submission, ensure that all pages of your solution are present and in order. Submit this PDF on Gradescope under `Assignment 2 - Written`. Please match all questions to their respective solutions (pages) on Gradescope. Questions not associated with any pages will be considered blank or missing and all questions need to be completed to receive full credit.

3. Questions

Q1 (30): Code

Implement the Viterbi algorithm (for pseudocode see Figure A.9 in the course textbook) using the `viterbi` function in `q1.py`. A class named `HMM` is provided in the template to assist with the implementation. You need to use this class in your implementation; however, you do not have to understand its inner workings to complete this assignment. The function of the `HMM` class is described in the `main` function. You will find further specifications in the comments so please read them carefully.
Supplementary material: `q1.py`

Q2 (40): Code + Written

Q2(a)(20): Code

(30%) Implement `get_unigram_counts` and `get_bigram_counts` functions as specified in `q2.py`. Both of these functions take as input a text string and convert it into a dictionary containing the lower case n-grams as keys and the respective frequencies as values. More instructions are provided in the comments so please read them carefully.
Supplementary material: `q2.py`

Q2(b)(20): Written

Download two different books of your choice (preferably from different authors to enable comparison) in `Plain Text UTF-8` format from Project Gutenberg.[1] Load these books as strings using Python's built-in `read()` function (learn more about it here[2]) and run your `get_unigram_counts` and `get_bigram_counts` functions on each book independently. Identify at least three similarities and three differences between the books, based on their n-gram frequencies (e.g., do both have the same most common unigram?). What information do these n-grams provide about the writing style of the author?
Supplementary material: `q2.py`

[1] `https://www.gutenberg.org/wiki/Main_Page`
[2] `https://www.w3schools.com/python/python_file_open.asp`

Q3 (30): Written

Given the following training documents and class labels (h = *healthy*, u = *unhealthy*), **compute all model parameters in log_{10} space for a multinomial Naive Bayes classifier using Maximum Likelihood Estimation (MLE) with Laplace smoothing**.

id	Training document	class
1	sanitizer masks	h
2	doorknobs crowds	u
3	masks distance	h
4	nightclubs crowds	u
5	sanitizer outdoors	h

Apply this model to the following test documents. What class will it assign to each document? What model/text characteristic(s) contributed to this label assignment?

id	Test document	class
6	sanitizer doorknobs	?
7	outdoors crowds	?

Show all model parameters and computations to get full credit.
Supplementary material: `NA`

4. Rubric

This assignment will be graded according to the rubric below. Partial points may be awarded for rubric items at the discretion of the course staff.

Q1 (30 points possible)	
(Autograded)	
Q2(a) (20 points possible)	
(Autograded)	
Q2(b) (20 points possible)	
Three n-gram similarities are clearly described	+7.5
Three n-gram differences are clearly described	+7.5
Inferences about writing style are sensible and clearly explained	+5
Q3 (30 points possible)	
Correct $p(h)$	+1
Correct $p(u)$	+1
Correct $p(\text{sanitizer}\|h)$	+1
Correct $p(\text{sanitizer}\|u)$	+1
Correct $p(\text{masks}\|h)$	+1
Correct $p(\text{masks}\|u)$	+1
Correct $p(\text{doorknobs}\|h)$	+1
Correct $p(\text{doorknobs}\|u)$	+1
Correct $p(\text{crowds}\|h)$	+1
Correct $p(\text{crowds}\|u)$	+1
Correct $p(\text{distance}\|h)$	+1
Correct $p(\text{distance}\|u)$	+1
Correct $p(\text{nightclubs}\|h)$	+1
Correct $p(\text{nightclubs}\|u)$	+1
Correct $p(\text{outdoors}\|h)$	+1
Correct $p(\text{outdoors}\|u)$	+1
Correct $p(h\|\text{sanitizer doorknobs})$	+2
Correct $p(u\|\text{sanitizer doorknobs})$	+2
Correct $p(h\|\text{outdoors crowds})$	+2
Correct $p(u\|\text{outdoors crowds})$	+2
Correct class assignment for "sanitizer doorknobs"	+1
Correct class assignment for "outdoors crowds"	+1
Key text characteristics leading to classification of "sanitizer doorknobs" are explained	+2
Key text characteristics leading to classification of "outdoors crowds" are explained	+2