CSCI 59000 NLP Spring 2023 Homework 3

February 9, 2024

Instructions

We will be using Canvas to collect your assignments. Please read the following instructions to prepare your submission.

1. Submit your solution in one zip file (<yourFirstName_LastName>.zip), which contains one pdf file and one folder. Your write-up must be in pdf, and your code must be in the folder named "code".
2. In your pdf file, the solution to each problem should start on a new page.
3. Latex is strongly encouraged to write your solutions, e.g., using Overleaf (https://www.overleaf.com/). However, scanned handwritten copies are also acceptable. Hard copies will not be accepted.
4. You may discuss the problems and potential directions for solving them with another student. However, you need to write your own solutions and code separately, and not as a group activity. Please list the students you collaborated with on your submission.

Problem 1 (10 points)

We are given the following text data. Each row represents a document which consists of two words

Word 1	Word 2	Label
blue	gloves	Y
blue	hat	Y
cautious	cat	Y
new	hat	Y
cautious	cat	Y
cautious	cat	N
cautious	hat	N
new	gloves	N
new	cat	N

Table 1: Text data for Problem 1

(a) Compute all the probabilities required to build a Naive Bayesian classifier. Ignore smoothing. Show your work.

(b) Using the probabilities from (a), compute the label for a new document "cautious gloves". Show your work.

Problem 2 (10 points)

Use the Penn Treebank tagset to tag each word in the following sentences. You may ignore punctuation.

(a) I take steps forward every day, no matter how small they may be.

(b) I solemnly swear that I am up to no good.

(c) Show what we truly are.

(d) The pen is mightier than the sword.

(e) Live in the now.

While tagging the sentences, were there any particular challenges?

Problem 3 (40 points)

We are given a Hidden Markov Model with the parameters below:

$S = \{N, M, V\}$ $K = \{$"*Patrick*", "*Cherry*", "*can*", "*will*", "*see*", "*spot*"$\}$

	π
N	0.7
M	0.1
V	0.2

Table 2: Initial probabilities in our HMM model

	N	M	V
N	0.2	0.3	0.5
M	0.4	0.1	0.5
V	0.8	0.1	0.1

Table 3: Transition probabilities in our HMM model

	Patrick	Cherry	can	will	see	spot
N	0.3	0.2	0.1	0.1	0.1	0.2
M	0	0	0.4	0.6	0	0
V	0	0	0.1	0.2	0.5	0.2

Table 4: Emission probabilities in our HMM model

Answer the following questions.

(a) Based on the model, how likely a sentence "Patrick can see Cherry" occur? Show your work.

(b) Based on the model, how likely a sentence "will Cherry spot Patrick" occur? Show your work.

(c) Based on the model, find the most likely tag sequence of "Patrick can see Cherry". Show your work.

(d) Based on the model, find the most likely tag sequence of "will Cherry spot Patrick". Show your work.

Now, please implement the Viterbi algorithm from scratch in Python 3. Using any third party library or already existing implementation is not allowed. Using Numpy is fine.

Make a file named `run_viterbi.py` where the main function calls the function named `viterbi()`. Arguments of the `viterbi` will be all the HMM parameter values (i.e., initial probabilities, transition probabilities, and emission probabilities) and a test sentence.

In the main function, call the `viterbi` function twice, first to compute (c), and then to compute (d).

(e) We will run your code using the command `python run_viterbi.py`. Write your code to print out mostly likely tag sequences for (c) and (d) with the probability for each state. You only need to print probabilities of the states in the most likely sequences. Print these to the screen.

(f) Are the most likely sequences you obtained by running your code the same as the sequences you obtained in (c) and (d)? If not, what would be the reason?

Problem 4 (40 points)

We will build a logistic regression classifier, using a well-known machine learning python package `scikit-learn (sklearn)`.

You will need to have `run_lr.py`. When executing the file using `python run_lr.py`, it needs to print your answers to (e) and (f) to the screen. (You need to include the answers in your pdf submission as well.)

First, download *polarity dataset v2.0* from this website. When uncompressing the file, you will find data in two folders: *pos* and *neg*. Yes, this is a dataset for sentiment analysis.

Sort the files in each folder in ascending order using the filenames. Move the last 200 samples in each folder to new folders (`pos_test`, `neg_test`), which will be our test data. Change the original folder names into `pos_train` and `neg_train`, respectively. Now your `pos_train` and `neg_train` have 800 samples each, and `pos_test`, `neg_test` have 200 samples each. The files in _train and _test folders must be different.

`pos` and `neg` will be the labels for the documents. When you read the documents in your code, your code will have to remember the label for each document.

We will only use adjectives as our features for the classifier.

Sklearn documentation

(a) The documents are already tokenized, and separated by whitespace. Using the python NLP package, `spaCy` (documentation), obtain POS tags. Using the POS tags, make a vocabulary of only adjectives by using the training data.

(b) Now make bag-of-word binary features only for those adjectives. You will have a feature vector with the size of the adjective vocabulary. Then, each document can be represented as a feature vector – if an adjective appears in the document, you set the feature value 1, otherwise 0. Represent all the documents in training set and test set as feature vectors.

(c) Write code for logistic regression using sklearn (documentation. Use your own feature representations from (b) for training and testing. Train on the training data.

(d) Add code for evaluation metrics. We will use f1 score, and you can use `sklearn.metrics.f1_score` to obtain f1 score.

(e) Apply your trained model on the training data. What is the f1 score (i.e., training accuracy)?

(f) Apply your trained model on the test data. What is the f1 score (i.e., test accuracy)?

(g) Discuss the results. Were the features effective for this classification? In either case (yes or no), why do you think it was the case? Any additional features you can think of? Describe one idea.