EECS595: Natural Language Processing
Homework 1, Fall 2023

Student Name: Rishikesh Ajay Ksheersagar — uniqname: RISHIKSH

Submission Guidelines

1. Please insert your student information in line 50 of this LaTeX file;
2. Please insert your answers between each pair of `\begin{solution}` and `\end{solution}`;
3. For programming problems (explicitly marked as `Programming`, please complete the python programs as instructed in `hw1.ipynb`.
4. Zip the files and submit to CANVAS. Checklist: `hw1.pdf`, `min_edit_dist.py`, `named_entity.py`, and `naive_bayes.py`.

Mutual Information and Divergence

Problem 1

Consider X and Y as two random variables. Let $p(x, y) = \Pr[X = x \wedge Y = y]$ and $p(x)$ and $p(y)$ be $\Pr[X = x]$ and $\Pr[Y = y]$, respectively. Show the steps to prove the following equation holds:

$$I(X;Y) = D_{KL}(p(x,y)||p(x)p(y))$$

Note: $I(X;Y)$ are mutual information of random variables X and Y,and D_{KL} stands for the Kullback–Leibler divergence.

Solution: We can find the relation between Mutual Information and KL Divergence as below:

$$\begin{aligned}
I(X;Y) &= H(Y) - H(Y|X) \\
&= -\sum_y p(y)\, log\, p(y) + \sum_{x,y} p(x)\, p(y|x)\, log\, p(y|x) \\
&= \sum_{x,y} p(x,y)\, log\, p(y|x) - \sum_y \left(\sum_x p(x,y) \right) log\, p(y) \\
&= \sum_{x,y} p(x,y)\, log\, p(y|x) - \sum_{x,y} p(x,y)\, log\, p(y) \\
&= \sum_{x,y} p(x,y)\, log\, \frac{p(y|x)}{p(y)} \\
&= \sum_{x,y} p(x,y)\, log\, \frac{p(y|x)\, p(x)}{p(y)\, p(x)} \\
&= \sum_{x,y} p(x,y)\, log\, \frac{p(x,y)}{p(y)\, p(x)}
\end{aligned}$$

Now, we know that

$$D_{KL}(p \,||\, q) = \sum_{x \,\epsilon\, X} p(x)\, log\left(\frac{p(x)}{q(x)}\right)$$

so,

$$D_{KL}\left(p(x,y) \,||\, p(x)\, p(y)\right) = \sum_{x,y} p(x,y)\, log\, \frac{p(x,y)}{p(y)\, p(x)}$$

Hence,

$$I(X;Y) = D_{KL}\left(p(x,y) \,||\, p(x)\, p(y)\right)$$

Minimum Edit Distance

Problem 2

In computational biology, comparing DNA sequences efficiently is a common problem. One way to measure the similarity between two sequences is by computing their Minimum Edit Distance (MED). The MED is the minimum number of insertions, deletions, and substitutions needed to transform one sequence into another.

For this problem, you are given two DNA sequences, where each sequence is represented by a string of the characters A, T, G, C. Please **write a python program** using **dynamic programming**, *i.e.*, for DNA sequences with length n, your implementation should have a time complexity of $O(n^2)$ and space complexity of no more than $O(n^2)$. Suppose the deletion and the insertion costs are "1" respectively and the substitution cost is "2".

For your convenience, a jupyter notebook `hw1.ipynb` and a skeleton python program `min_edit.py` are provided for you. You may use it as needed.

1. (Programming) Implement `med_dp_top_down(str1, str2)` and `med_dp_bottom_up(str1, str2)` in `min_edit_dist.py`. Both implementations have a time complexity of $O(n^2)$ and space complexity of no more than $O(n^2)$. It's possible to reduce the space complexity to $O(n)$ in the bottom up. In fact, to receive full credits, the bottom-up version should have a space complexity of $O(n)$. *(hint: check the function return)*

2. (Written) This is the exact same MED algorithm in the field of NLP, where it serves as a naive tool for tasks like spelling correction, text similarity assessment, and fuzzy search. Figure out whether *manager* is closer to *manner* or to *mangoes*. What is the minimum edit distance in each case? What are the steps (list one of the minimum editing possibilities)?

Solution: 2. Levenshtein minimum edit distance, also known as edit distance, measures the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. It's a way to quantify the similarity or dissimilarity between two strings by counting the minimum number of operations needed to make them identical.

Checking the *MED* between *MANAGER* and *MANGOES* below-

		M	A	N	A	G	E	R
	0	1	2	3	4	5	6	7
M	1	0	1	2	3	4	5	6
A	2	1	0	1	2	3	4	5
N	3	2	1	0	1	2	3	4
G	4	3	2	1	2	1	2	3
O	5	4	3	2	2	2	2	3
E	6	5	4	3	3	3	2	3
S	7	6	5	4	4	4	3	4

The *MED* between *MANAGER* and *MANGOES* is 4. One way to get *MANGOES* from *MANAGER* is to DELETE 2nd "A" from *MANAGER*, INSERT an "O" right after and lastly SUBSTITUTE "R" from *MANAGER* to "S".

Checking the MED between MANAGER and MANNER below-

		M	A	N	A	G	E	R
	0	1	2	3	4	5	6	7
M	1	0	1	2	3	4	5	6
A	2	1	0	1	2	3	4	5
N	3	2	1	0	1	2	3	4
N	4	3	2	1	2	3	3	4
E	5	4	3	2	2	3	3	3
R	6	5	4	3	3	3	3	3

The *MED* between *MANAGER* and *MANNER* is 3. One way to get *MANNER* from *MANAGER* is to DELETE the 2nd "A" from *MANAGER* and then SUBSTITUTE the "G" in *MANAGER* with an "N".

Hence, we can say that *MANAGER* is closer to *MANNER* than *MANGOES*.

Regular Expression

Problem 3

In a textual document from logs, emails, or web pages, certain textual strings have special semantic meanings. These could be related to various named entities such as IP addresses, time stamps, URLs, email addresses, or transaction amounts. These entities are crucial in the field of cybersecurity for monitoring transactions, detecting fraud, and tracing malicious activities.

We have seen in the class that regular expressions can be used to match a text string to identify these named entities related to that text string. For example: `/$[0-9]+\.[0-9][0-9]/` can be used to match any string indicating some dollar amount, *e.g.*, `$149.99`. Once a match is found, the system can assign a special name (*e.g.*, `DOLLAR_AMOUNT`) to the string (*i.e.*, `$149.99`).

Please **write a python program** where you will specify regular expressions to identify the following types of named entities: `IP_ADDRRESS`, `TIME`, `DATE`, `EMAIL_ADDRESS`, `WEB_ADDRESS`, `DOLLAR_AMOUNT`. Define a function called `cyber_ner(input)` which will output the sentence that replaces the special strings in `input` to their recognized named entities.

For example, if `input` is:

> "Email from `john.doe@example.com` at `15:30` on `07/25/2023` regarding the transaction of `$1,200`. Log shows an incoming connection from IP `192.168.1.1`. Please review at

`https://cybersec.com`."

`cyber_ner(input)` should return:

"Email from `EMAIL_ADDRESS` at `TIME` on `DATE` regarding the transaction of `DOLLAR_AMOUNT`. Log shows an incoming connection from IP `IP_ADDRESS`. Please review at `WEB_ADDRESS`."

We will not give you any training data. You should use your knowledge to come up with the appropriate regular expressions. Try to cover as many possibilities as you can think of. Your program will be tested on a small set of testing examples.

For your convenience, a jupyter notebook `hw1.ipynb` and a skeleton python program `named_entity.py` are provided for you. You may use it as needed.

1. (Programming) Please submit your python program in `named_entity.py`.
2. (Written) What are the potential problems with this approach to named entity recognition? Please name at least two issues.

Solution: Named Entity Recognition (NER) using regular expressions has several limitations and problems that can affect its accuracy and effectiveness. Here are some of the key issues I noticed while working on this problem:

1. *Inability to handle variability*: Named entities often exhibit significant variability in their forms. For instance, the date can take up multiple formats such as DD/MM/YYYY, MM/YYYY, DD MMM YYYY, DD¡st/nd/rd/th¿ MMM YY, MonthName Day Year, and so on. Constructing regex patterns to account for all possible variations is challenging, and the resulting expressions can become complex and hard to maintain.
2. *Handling Ambiguity*: Named entities often have multiple interpretations or may belong to multiple categories. For instance, 12/2020 can represent a valid month-year and hence a date but it can also be a random fraction which has nothing to do with date. Regex patterns may not handle such ambiguity well as it also needs to have some knowledge of context to accurately flag such cases as one entity or another.
3. *Scalability and Maintenance Challenges*: Maintaining and scaling a regex-based NER system can be labor-intensive. As the volume and diversity of text data increase, so does the complexity of regex patterns required to capture all relevant entities accurately. Periodic updates are necessary to adapt to evolving language usage, new entity types, and changing data formats. This process can be time-consuming and error-prone.

Mutual Information

Problem 4

Please write a script to analyze a provided dataset to answer the following questions.

We will use a standard movie dataset for sentiment analysis, provided on Canvas in `hw1_data.zip`. In the `training` and `testing` directories, there are two subdirectories `pos` (which consists of documents labeled as having positive sentiment) and `neg` (which consists of documents with negative sentiment). In the "pos" subdirectory, each text has a class label $c = 1$. In the "neg" subdirectory, each text has a class label $c = -1$. Let w be a binary random variable so that $w = 1$ if w appears in a review, and $w = 0$ otherwise.

- Please use the **training** set.
- Treat each token as a word and do not further process words.
- Treat punctuations as words.

For your convenience, a jupyter notebook `hw1.ipynb` is provided for you. You may use it as needed. Answer the questions below:

1. (Written) For each word w you need to collect four numbers: the number of reviews in which w occurs and $c = 1$; w doesn't occur and $c = 1$; w occurs and $c = -1$; w doesn't occur and $c = -1$. List these four numbers for the following four words "the, love, awful, movie" respectively.

2. (Written) Write down the formula of mutual information $I(w; c)$.

3. (Written) Compute $\log_2 I(w; c)$ for all word types. Make sure you use log base 2 to get bits. What is the mutual information for the words "the, love, awful, movie" respectively?

4. (Written) List the top five words with the highest mutual information and five words with the lowest mutual information.

5. (Written) Browse the list of words with their corresponding mutual information. Find 3 words with unexpected mutual information and explain why they are unexpected. It is up to you to define what is "unexpected". There is no standard answer to this question.

Solution: **Answer 1**: the number of reviews in which w occurs and $c = 1$; w doesn't occur and $c = 1$; w occurs and $c = -1$; w doesn't occur and $c = -1$ for "the", "love", "awful", "movie" are as follows:

1. *the*

	w = True	w = False
c = 1	554	0
c = -1	552	0

2. *love*

	w = True	w = False
c = 1	197	357
c = -1	142	410

3. *awful*

	w = True	w = False
c = 1	9	545
c = -1	52	500

4. *movie*

	w = True	w = False
c = 1	442	112
c = -1	453	99

Answer 2: The formula for Mutual Information, I(w;c) can be written as:

$$I(w;c) = \sum_{w} \sum_{c} p(w,c)\, log_2 \frac{p(w,c)}{p(w)\, p(c)}$$

Answer 3:

1. The Mutual Information for the word " the " is = 0.0
2. The Mutual Information for the word " love " is = 0.0082
3. The Mutual Information for the word " awful " is = 0.0231
4. The Mutual Information for the word " movie " is = 0.0006

Answer 4:

Top 5 words with highest mutual information:

1. ('bad', 0.035697866509854584)
2. ('worst', 0.03324788021778111)
3. ('?', 0.031235346351495802)
4. ('wasted', 0.027889385980003296)
5. ('wonderfully', 0.025435305898912557)

Five words with lowest mutual information (including 0s):

1. ('.', 0.0)
2. ('and', 0.0)
3. ('the', 0.0)
4. ("'98", 4.2732265895838215e-09)
5. ("'dark", 4.2732265895838215e-09)

Five words with lowest mutual information (excluding 0s):

1. ("'98", 4.2732265895838215e-09)
2. ("'dark", 4.2732265895838215e-09)
3. ("'deep", 4.2732265895838215e-09)
4. ("'die", 4.2732265895838215e-09)
5. ("'just", 4.2732265895838215e-09)

Answer 5:

1. Firstly, I noticed that the mutual information for words like *also*, *both*, *others*, *quite* is quite high. This is counter-intuitive as these are generic words and I expected these to have lower mutual information.
2. "*?*" is the "word" with 3rd highest mutual information. This is unexpected as one of the most common punctuation is expected to have a much lower mutual information.
3. The mutual information for *man* is quite higher than the mutual information for *woman* but the mutual information for *actor* is lower than the mutual information of *actress*. This is a bit unexpected.
4. Many sentiment defining words like *unmemorable*, *unwelcome*, among others have very low mutual information which is unexpected. I believe this is happening because we are calculating the mutual information just based on the data we have handy and not using a larger and more exhaustive vocabulary, I mean, the mutual information for these words is low in our data of 1106 reviews but if we look at more exhaustive and larger datasets, it might tell us otherwise.

Naïve Bayes Classifier

Problem 5

Please implement a simple naïve Bayes classifier for sentiment analysis. We will use the same movie dataset in problem 4.

You will use the **training** data to learn model parameters using features of your choice (*e.g.*, simplest features are bag of words). Your trained model will be applied to the **testing** data to predict the label for each testing document. Ground-truth labels are also provided in the testing data, so you should use them to calculate the performance of your model using *accuracy*.

Your program should consist of three modules:

- A **training module** that learns the parameters
- A **testing module** that applies the learned model to make predictions on the testing data
- An **evaluation module** that calculates the accuracy of the predictions made by the model

The README file provided with the dataset provides some background on the kind of features you may consider. For your convenience, a skeleton Python program is provided for you in `naive_bayes.py`. You may use it as needed.

1. (Programming) Please submit your python program in `naive_bayes.py`. In this assignment, you have the freedom to use any pre-processing you see fit and any features of your choice.
2. (Written) Explains your designs and results, *i.e.*, the set of features you have applied, and report your results in this latex file.

Solution:

- Design and Steps
  1. Read the Training data
  2. Cleaning:
     (a) Removed the URL links
     (b) Removed the special symbols
     (c) Removed the new line characters
     (d) NOTE: Didn't remove stop words as it was reducing the accuracy of the model
     (e) Performed Lemmatization by using WordNetLemmatizer from nltk.stem
  3. Features used:
     (a) Post cleaning the data, used CountVectorizer from sklearn.feature_extraction.text with tokenizer as word_tokenize from nltk and analyzer set to "words".
     (b) Generated a set of Bi-grams by passing ngram_range=(2,2) in the Vectorizer.
     (c) This helped generate a training set - 1106x298206 sparse matrix with 681020 stored elements in Compressed Sparse Row format

- Results:
    1. Precision = 0.8354
    2. Recall = 0.9429
    3. Accuracy: 87.857%
    4. F-1 Score: 0.8859
    5. Confusion Matrix:

	Predicted Negative	Predicted Positive
Actual Negative	57	13
Actual Positive	4	66