Assignment 2

CS 421: Natural Language Processing

1 Introduction

In this assignment, you will learn about the implementation of two basic classifiers, the Naive Bayes Classifier and the Logistic Regression Classifier. All the code must be yours and **use of libraries like `scikit-learn` is prohibited unless stated otherwise.**

Naive Bayes is a classifier where for a document d, out of all classes $c \varepsilon C$ it returns the class $\hat{c}$ which has the maximum posterior probability given the document. It is a **generative** model which calculates the likelihood $p(d|c)$ and the prior $p(c)$ to predict the class instead of calculating $p(c|d)$ directly. By contrast, a **discriminative** model, like **Logistic regression**, attempts to directly compute the posterior $p(c|d)$ by learning boundaries in the data space.

You will also learn to implement the **Gradient Descent** algorithm for binary logistic regression. The objective of the algorithm is to minimize the loss defined by the learned boundary to update the parameters of the model. It tries to find the local minima of the loss function as the loss will be lowest there in the given region.

2 Instructions

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

Code

This section needs to be completed using Python 3.6+. You will also require following packages:

- pandas
- numpy
- NLTK or SpaCy
- scikit-learn

If you want to use an external package for any reason, you are required to get approval from the course staff prior to submission.

3 Questions

Q1. Naive Bayes: Code [25]

In this question, you will learn to build a Naive Bayes Classifier for the binary classification task.

1. Dataset: "Financial Phrasebank" dataset from HuggingFace.[1] To load the data, you need to install library "`datasets`" (`pip install datasets`) and then use `load_datset()` method to load the dataset. You can find the code on the link provided above.
2. The dataset contains 3 class labels, neutral (1), positive (2), and negative (0). Consider only positive and negative samples and ignore the neutral samples. Use 80% of the samples selected randomly to train the model and the remaining 20% for the test.
3. Clean the dataset with the steps from the previous assignment and build a vocabulary of all the words.
4. Compute the prior probability of each class

   $$p(c_i) = \frac{count(c_i)}{N}$$

   Here, $count(c_i)$ is the number of samples with class c_i and N is the total number of samples in the dataset.
5. Compute the likelihood $p(w_i|c)$ for a all words w_i and all classes c with following equation:

   $$p(w_i|c) = \frac{count(w_i, c) + 1}{|V| + \sum_{w \varepsilon V} count(w, c)}$$

   Here, the $count(w_i, c)$ is the frequency of the word w_i in class c while $\sum_{w \varepsilon V} count(w, c)$ is the frequency of *all* the words in the class c. Laplace smoothing is used to avoid zero probability in the case of a new word.
6. For each sample in the test set, predict class c_{NB} which is the class with the highest posterior probability. To avoid underflow and increase speed, use log space to predict the class as follows:

   $$c_{NB} = \underset{c \varepsilon C}{\text{argmax}}(\log p(c) + \sum_{w_i \varepsilon V} \log p(w_i|c)) \tag{3.1}$$

7. Using the metrics from `scikit-learn` library[2], calculate the accuracy and macro-average precision, recall, and F1 score, and also provide the confusion matrix on the test set.

Q2. Logistic Regression: Code [25]

In this task, you will learn to build a Logistic Regression Classifier for the same "Financial Phrasebank" dataset. *Bag of Words* model will be used for this task.

1. Use 60% of the data selected randomly for training, 20% selected randomly for testing and the remaining 20% for validation set. Use classes 'positive' and 'negative' only. Perform the same cleaning tasks on the text data and build a vocabulary of the words.

[1] Link to the dataset.
[2] Link to the scikit-learn documentation for metrics.

2. Using `CountVectorizer`[3], fit the cleaned train data. This will create the *bag-of-words* model for the train data. Transform test and validation sets using same `CountVectorizer`.
3. To implement the logistic regression using following equations,

$$z_i = W.x_i$$

$$\hat{y}_i = \sigma(z_i)$$

   we need the weight vector W. Create an array of dimension equal to those of each x_i from the CountVectorizer.
4. Apply above equations over whole training dataset and calculate $\hat{y}$ and cross-entropy loss $\mathcal{L}_{CE}$ which can be calculated as

$$\mathcal{L}_{CE} = -y \log \hat{y} + (1 - y) \log(1 - \hat{y})$$

5. Now, update the weights as follows:

$$W_{t+1} = W_t - \alpha(\hat{y}_i - y_i).x_i$$

   Here, $(\hat{y}_i - y_i).x_i$ is the gradient of *sigmoid* function and $\alpha = 0.01$ is the learning rate.
6. Repeat step 4 and step 5 for 500 iterations or epochs. For each iteration, calculate the cross-entropy loss on validation set.
7. Calculate the accuracy and macro-average precision, recall, and F1 score and provide the confusion matrix on the test set.
8. Experiment with varying values of $\alpha \in (0.0001, 0.001, 0.01, 0.1)$. Report your observations with respect to the performance of the model. You can vary the number of iterations to enhance the performance if necessary.

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

4 Rubrics

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 Naive Bayes: (25 points possible)	
Q 1.1 - Q 1.3	+5
Q 1.4	+5
Q 1.5	+5
Q 1.6	+5
Q 1.7	+5
Q2 Logistic Regression: (25 points possible)	
Q 2.1 - Q 2.2	+5
Q 2.3 - Q 2.4	+5
Q 2.4 - Q 2.7	+10
Q 2.8	+5