

CS 445 Natural Language Processing

**Project 1: Text Classification**

**Due Date: May 10, 23:55**

In this project, you are expected to develop a text classification system for English customer reviews in two parts. In the first part you will experiment with the non-neural algorithms;

* *Naïve Bayes*
* *Logistic Regression*

In the second part, you will experiment with a neural-based algorithm;

* *Convolutional Neural Networks (CNN)*

**You will be given a Colab notebook and you are supposed to implement these models for binary and multiclass classification with the shared dataset**. **For these different models, you will try different hyper-parameters. You will write your findings, results, and interpretations at the end of the notebook.**

You can find the notebook here:

https://colab.research.google.com/drive/1x3vJ-3--nSt4BT2H7fwOuY4S25GlHL4b?usp=sharing

Please make a copy of this notebook and work on your own copy.

**Dataset**

The dataset provided to you is a part of the customer reviews data collection. It consists of customer reviews from [Yelp](https://www.kaggle.com/datasets/yelp-dataset/yelp-dataset). The documents (customer reviews) were labeled based on their sentiments.

You are provided with a train and test dataset. These datasets consist of two columns;

* ***text***: This column consists of raw text of customer reviews.
* ***label***: This column consists of the multiclass label of sentiments of a given customer review. There are 5 levels of sentiments [1,2,3,4,5]

You are supposed to create two different datasets from the given train and test sets. The first dataset will be a multiclass dataset which has 5 classes, as the original dataset, and the second dataset will be a binary dataset that has only 2 classes. You will create your binary dataset from the original dataset by removing *class 3*, and mapping *classes 1, 2* to 0, and *classes 4, 5* to 1.

**After building your multiclass and binary datasets, please report the class distributions from both train and test sets of each dataset.**

**Implementation Part**

In this project, you are expected to use Google Colab since some of the classification algorithms require more computation. You will do all your implementations on ‘***Project 1 Notebook.ipynb****’* file and submit that file with the expected outputs. The first version of the notebook will only contain relevant cells for *Naïve Bayes* and *Logistic Regression,* the notebook will be updated with the cells for the *Convolutional Neural Network* in the upcoming weeks.

In your implementations, you will do the followings:

* *Preprocessing:*

In this part, after manipulating the original dataset in order to create the multiclass and binary datasets, you can choose to do some preprocessing steps like lowercasing tokens, removing stopwords, stemming, etc. These may be useful for some traditional classification algorithms.

* *Classification:*

In the first part of the project, you will use Naïve Bayes and Logistic Regression classifiers. They use different hyper-parameters. You need to understand these and fine-tune them. You can use the Pipeline and GridSearchCV functions of sklearn for this purpose. To make sure everyone is working with the same hyperparameter space, the parameters will be mentioned below.

1. **Naïve Bayes (NB)**: You will use a pipeline of three components. The first component will be a method you will choose that can vectorize your text input in order to feed it into a classifier. In other words, you can check a module of sklearn for implementing the term-weighting. You should try different *ngrams,* up to 3-grams. For a term to be valid it should exist in at least N documents. You should try 100, 500, and 1000 as your N value. Since your term-weighting method will return a matrix with float values, you should use an appropriate NB method. There is no need to try different parameters for NB, keep the default version. Lastly, a middle component, called DenseTransformer, is implemented in the notebook. You should use this to convert your sparse matrix into a dense matrix.
2. **Logistic Regression (LR)**: There should be only two components of your **LR** pipeline. The first component is the same described above for NB; term-weighting module. Please use the same hyperparameter space for the term-weighting module for the LR pipeline as well. The second component will be the LR classifier. You should set the random state for the classifier to 22. You should also try different values for regularization distribution between L1 and L2 regularization. Please find the correct hyperparameter and try ratios 0.0, 0.5, and 1.0.

You will run the GridSearchCV for both binary and multiclass datasets. Your algorithm should try to maximize the F1-Macro score with 5-fold cross-validation.

You will also report the max, min, mean, and standard deviation of scores for each parameter group (Check the cv\_results\_ of GridSearchCV). Other details are available in the notebook.

In the second part, you will use a Convolutional Neural Network (CNN) classifier. You can use the keras library for implementing your CNN classifier. You will try different hyper-parameters for word embeddings and the CNN algorithm. The base model should consist of an embedding layer followed by at least a Conv and Dense layer. For the embeddings, you should at least try the following three embedding strategies;

* Randomly initialized word embeddings
* Word embeddings trained from scratch with gensim
* Pretrained word embeddings from gensim.api (you can pick one)

For the Conv layer, you should try at least two kernel sizes and filter sizes. For the Dense layer, you should try at least two different hidden layer sizes. All other details are dependent on your choice. You should apply the aforementioned steps for both binary and multi-class classification. Additional to the requirements mentioned, you can try different architectures and embedding strategies. There are much fewer details on the implementation of this part in the notebook. You can find quite different ways of doing every component of CNN part. Please do not forget to add a reference to the pages where you have taken a piece of code, etc. Please report all your findings.

* *Evaluation*:

You are going to report the F1 and Accuracy scores. You will also print the confusion matrix. You will use these to discuss and compare the approaches in your report. Do not just state the obvious please elaborate on the results based on what you have learned in the class.

Make sure that your ‘***Project 1 Notebook.ipynb****’* is well commented. After running all the cells, export the .*ipynb, .py,* *html* output of your notebook, and put a link to your Colab notebook in a txt file, upload that as well. In the end, you should submit 4 files.

You can use the popular/standard python packages. In case you are not sure of a particular library, please ask the instructor or TAs.

You are expected to implement this project on your own. Your scripts will be analyzed by using state-of-the-art tools for any type of plagiarism.

**Report:**

In your report, you are going to summarize your approach and your findings. Discuss what is working and what is not. Especially with CNN, you need to discuss the effects of architecture and word embeddings on performance in detail. You will write your findings at the end of the Colab, section named “My Report”.

**Your notebook will be evaluated with state-of-the-art anti-cheating programs.**

All files should be under the same directory (named as your student ID, only the 5 digit numbers), which you will zip and submit.

Submission Instructions:

* You will submit this project via SUCourse.
* Please check the slides for the late submission policy.
* You can resubmit your project (until the deadline) if you need to.
* Please read this document again before submitting your solution.
* After submitting, you should download your submission to a different path to double-check whether everything is in order.
* Please do your assignment individually, do not copy from a friend or the Internet. Plagiarized assignments will receive -100.