ASSIGNMENT 1 - USER RATING

January 31, 2023

1 Motivation

The purpose of this assignment is to provide an opportunity for students to gain experience in text categorization utilizing Machine Learning techniques.

2 Problem Statement

The objective of this assignment is to develop a text categorization system that predicts user ratings based on the review text they provide. The system will receive the review text as input and generate the corresponding user rating as output.

3 Training Data

This assignment will utilize a training dataset of reviews referred to as `train.csv`. The following code serves as an example for reading the dataset:

[2]:
```
import pandas as pd

train_df = pd.read_csv('train.csv', header=None)
```

[3]:
```
train_df.head()
```

[3]:

	0	1
0	Definitely runs long, wore 4.5 inch heels and …	5
1	I think it's a little big/long in the torso.	5
2	This dress is amazing! It has a built in like…	5
3	didn't even need a bra and I am a 34DD, was so…	5
4	I wore this to my birthday dinner and loved it…	4

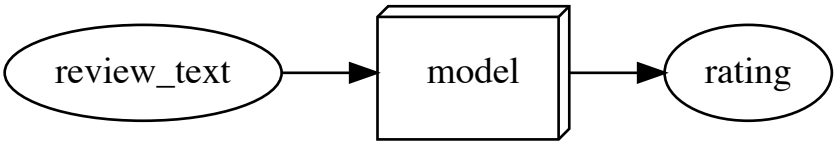
The `train.csv` file contains two columns: `review text` and `rating`. The file consists of 134,704 rows. The `review text` column includes the textual review provided by the user, while the `rating` column features the corresponding rating, which is represented as a number from 1 to 5.

4 Task

It is required to construct a classifier that, given a user review text, predicts the corresponding user rating. This part of the assignment requires the development of a non-neural classifier, which can

be achieved through various methods such as Naïve Bayes, logistic regression, nearest neighbor, rule/lexicon-based systems, support vector machines, and so on.

[28]:



As a starting point, a baseline algorithm can be established by utilizing all words as features and training a classifier, such as Naïve Bayes or logistic regression. Before proceeding, it is recommended to thoroughly consider the problem at hand. To enhance the performance of the system beyond the baseline, several improvement strategies are outlined as follows:

1. It is important to note that only a single set, rather than separate training and validation sets, will be provided. K-fold cross-validation is suggested, but not mandatory, as a method of evaluating the performance of the model. For further understanding, the concept of k-fold cross-validation can be researched independently and questions can be raised through discussion forums. Alternatively, the data can be divided into custom train and dev splits, and the model can be trained accordingly.
2. To address the class imbalance in the data, various sampling frequencies for different classes can be tested.
3. Regularization techniques, such as L1/L2 regularization in logistic regression or adjusting the regularization parameter in support vector machines, can be explored to further improve performance.
4. Consider working with the features by using techniques such as lemmatization, removal of stop words and infrequent words, and tf-idf weighting.
5. Incorporating bigrams or trigrams in addition to unigrams can be explored as a potential improvement strategy.
6. New features, such as the pos-tag of each word or the presence of capitalization, can be defined and utilized.
7. Off-the-shelf tools, such as named entity recognition, may be utilized as long as their applicability to the given problem is thoroughly considered. It is essential to confirm the non-neural nature of the tools used by thoroughly reviewing the documentation. For example, the Natural Language Toolkit (NLTK) offers only rule-based POS taggers, while SpaCy provides both rule-based and neural-based options.
8. The utilization of trained word embeddings or neural models is strictly prohibited for the task at hand. The use of pre-trained neural models, such as those for named entity recognition, is also not permitted.

5 Methodology and Experiments

The methodology for splitting the data into train and dev sets, as well as the ratio of split, is left to the discretion of the individual. It is recommended to perform k-fold cross-validation, with a value of k chosen by the individual. The best hyperparameters should be determined and set in the final training script to be submitted.

The validation accuracy, or k-fold validation accuracy, achieved must be reported. It is crucial to ensure that the code is deterministic in nature, as the results will be replicated by running the training code. The test data will not be released and will be held by the authorities.

It is important to document the performance of the baseline system as improvements are made. Error analysis should be performed on a subset of the data, and the reasons for the model's mistakes should be analyzed. This will help guide in selecting the next feature or model component to add.

Given the moderately large size of the dataset, it is advised to conduct smart grid searches instead of a plain grid search, as the latter may be too resource-intensive.

6 Evaluation Metric

We will assess the performance of your prediction output using micro-F1 and macro-F1 scores. Please familiarize yourself with these evaluation metrics. During validation, you may choose to apply a suitable combination of micro-F1 and macro-F1, such as a weighted sum or average. Both micro-F1 and macro-F1 will be evaluated on the test data and will be given equal importance in the grading process. The code we will use for evaluation is provided below.

[29]:
```python
import pandas as pd
from sklearn.metrics import f1_score

def evaluate(gold_file, pred_file):
    y_true = pd.read_csv(gold_file, header=None)
    y_pred = pd.read_csv(pred_file, header=None)

    f1_micro = f1_score(y_true, y_pred, average='micro')
    f1_macro = f1_score(y_true, y_pred, average='macro')

    return (f1_micro+f1_macro)/2
```

7 Test Format

Your prediction program must accept a set of user reviews in the same CSV format as the training data, consisting of a single column for the review text. The output of your testing script must be a file containing predicted ratings ranging from 1 to 5, with each prediction corresponding to a single review and appearing on a new line. A sample submission is provided for reference. The format is as follows:

[4]:
```python
pd.read_csv("sample_input.csv", header=None).head()
```

[4]:

	0
0	I can't say anything bad about this dress. It …
1	Overall I enjoyed the dress, however I did fee…
2	This is my favorite designer and love almost a…
3	I don't know if I have ever gotten more compli…
4	This dress was amazing!!! It fit perfectly and…

[5]:
```
pd.read_csv("sample_output.csv", header=None).head()
```

[5]:

	0
0	4
1	4
2	5
3	5
4	5

The deadline for submission is 13^{th} February 2023, 11:55 PM. Submit your code on Moodle in a .zip file named in the format `EntryNo`.zip (for example 2018CS50404.zip). Make sure that when we run "unzip yourfile.zip", a new directory is created with your entry number (in all caps). In that directory, the following files should be present. There can be more files for helper functions/utilities, but we are only concerned about the following:

- environment.yml
- train.sh
- test.sh
- writeup.txt
- link.txt

Kindly ensure that all your code is executed within an Anaconda environment, which can be created using the following command:

```
conda create -n EntryNo python=3.8
```

Please note that Python version 3.8 will be used for your evaluation. It is imperative that you do not include the dataset or any trained model files in your submission. Instead, kindly upload your model file to Owncloud and ensure that edit permissions are granted. The file name should be in the format `EntryNo.model` (e.g. `2018CS50404.model`). The download link for the model file should be provided in the first line of the `link.txt` file. Non-compliance with these requirements will result in penalties.

Your code will be executed as follows:

1. `conda env create --name EntryNo --file=environments.yml` (This will install the necessary packages to set up your environment.)
2. `./train.sh <input path to train.csv> trained_model` (This will create a model file named trained_model in the present directory.)
3. `./test.sh trained_model <input path to test.csv> output.csv` (This will generate an output predictions file named output.csv in the present directory.)

Kindly note that all commands mentioned above will be executed in an automated script. Any error(s) encountered during the execution of the script will result in a 20% reduction of the total

credit awarded. Hence, it is crucial that you adhere to the specified requirements.

The "output.csv" file should contain a sequence of ratings, with one rating per line, corresponding to the text reviews of users in the "test.csv" file. The number of lines in the "output.csv" file must be equal to the number of text reviews of the user in the "test.csv" file.

In the "writeup.txt" file, the first line should indicate the names of all students with whom you discussed or collaborated regarding this assignment. If you did not engage in any form of collaboration, kindly state "None". You are also welcome to provide an overview of your code, though this is optional.

It is important to note that the training process must be completed within 2 hours, using a single HPC CPU. As this is a non-neural assignment, usage of a GPU is not permitted. If the training process exceeds the time limit, significant penalties will apply.