CSCI544: Homework Assignment №1

Due on Jan 25, 2023 (before class)

This assignment gives you hands-on experience with text representations and the use of text classification for sentiment analysis. Sentiment analysis is extensively used to study customer behaviors using reviews and survey responses, online and social media, and healthcare materials for marketing and costumer service applications. The assignment is accompanied with a Jupyter Notebook to structure your code. Please submit:

1. A PDF report which contains answers to the questions in the assignment along with brief explanations about your solution. Please also print the completed Jupyter Notebook in PDF format and merge it with your report. (just submit one PDF file by merging your written answer and the completed Jupyter notebook). On your completed Jupyter notebook, please print the requested values, too.

2. You also need to submit an executable .py file which when run, generates the requested numerical outputs in the assignment as listed at the end of the assignment description. We need the .py file to check overlap between codes to detect plagiarism. Please include the Python version you use.

The libraries that you will need are included in the HW1.ipynb file. You can use other libraries as far as they decently similar to the ones included in the HW1.ipynb file. At the beginning of the .py file, add a read command to read the data.tsv file as the input to your .py file from the current directory.

1. Dataset Preparation (10 points)

We will use the Amazon reviews dataset which contains real reviews for jewelry products sold on Amazon. The dataset is downloadable at:

`https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Beauty_v1_00.tsv.gz`

(a)

Read the data as a Pandas frame using Pandas package and only keep the Reviews and Ratings fields in the input data frame to generate data. Our goal is to train sentiment analysis classifiers that can predict the rating value for a given review.

We create a three-class classification problem according to the ratings. The original dataset is large. To this end, let ratings with the values of 1 and 2 form class 1, ratings with the value of 3 form class 2, and ratings with the values of 4 and 5 form class 3. To avoid the computational burden, select 20,000 random reviews from each rating class and create a balanced dataset to perform the required tasks on the downsized dataset. Split your dataset into 80% training dataset and 20% testing dataset. Note that you can split your dataset after step 4 when the TF-IDF features are extracted.

Follow the given order of data processing but you can change the order if it improves your final results.

2. Data Cleaning (20 points)

Use some data cleaning steps to preprocess the dataset you created. For example, you can use:

- convert all reviews into lowercase.
- remove the HTML and URLs from the reviews
- remove non-alphabetical characters
- remove extra spaces
- perform contractions on the reviews, e.g., won't → will not. Include as many contractions in English that you can think of.

You can use other cleaning procedures that can help to improve performance. You can either use Pandas package functions or any other built-in functions. Do not try to implement the above processes manually.

In your report, print the average length of the reviews in terms of character length in your dataset before and after cleaning (to be printed by .py file).

3. Preprocessing (20 points)

Use NLTK package to process your dataset:

- remove the stop words
- perform lemmatization

In your report and the .py file, print the average length of the reviews in terms of character length in before and after preprocessing.

4. Feature Extraction (10 points)

Use sklearn to extract TF-IDF features. At this point, you should have created a dataset that consists of features and labels for the reviews you selected.

5. Perceptron (10 points)

Train a Perceptron model on your training dataset using the sklearn built-in implementation.

Study about generalizations of Precision, Recall, and f1-score in multi-class situations. Report Precision, Recall, and f1-score per class and their averages on the testing split of your dataset. These 12 values should be printed in separate lines by the .py file.

6. SVM (10 points)

Train an SVM model on your training dataset using the sklearn built-in implementation. Report Precision, Recall, and f1-score per class and their averages on the testing split of your dataset. These 12 values should be printed in separate lines by the .py file.

7. Logistic Regression (10 points)

Train a Logistic Regression model on your training dataset using the sklearn built-in implementation. Report Precision, Recall, and f1-score per class and their averages on the testing split of your dataset. These 12 values should be printed in separate lines by the .py file.

8. Multinomial Naive Bayes (10 points)

Train a Multinomial Naive Bayes model on your training dataset using the sklearn built-in implementation. Report Precision, Recall, and f1-score per class and their averages on the testing split of your dataset. These 12 values should be printed in separate lines by the .py file

Note 1: For questions 5-8, part of grading is based on being competitive. For each question, we will sort the computed average precision values across the class. For each question, the top 40% will receive full credit. The next 30% will loose 1 point, and the bottom 30% will lose 2 points. We have this grading scheme to motivate you to explore ideas for increasing your performance values.

Note 2: To be consistent, when the .py file is run, the following should be printed, each in a line:

- Average length of reviews before and after data cleaning (with a comma between them)

- Average length of reviews before and after data preprocessing (with comma between them)

- Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for Perceptron (with comma between the three values)

- Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for SVM (with comma between the three values)

- Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for Logistic Regression (with comma between the three values)

- Precision, Recall, and f1-score for the testing split in 4 lines (in the order of rating classes and then the average) for Naive Bayes (with comma between the three values)

Note that in your Jupyter notebook, print the Precision, Recall, and f1-score for the above models in separate lines and in .py file in separate lines.