CSCI544: Homework Assignment №1

Due on September 9, 2021 (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 a PDF report which contains answers to the questions in the assignment and also print the completed Jupyter Notebook in PDF format (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. Additionally, you also need to submit an executable .py file which when run, generates the requested numerical outputs in the assignment. 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.

1. Dataset Prepration (10 points)

We will use the Amazon reviews dataset which contains real reviews for kitchen products sold on Amazon. The dataset is downloadable at:
`https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Kitchen_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 sentiment (positive/negative) for a given review.

Include three sample reviews in your report along with corresponding ratings. Also, report the statistics of the ratings, i.e., how many reviews received 1 ratings, etc.

We create binary labels using the ratings. We assume that ratings more than 3 demonstrate positive sentiment (mapped to 1) and rating less than or equal 2 demonstrate negative sentiment (mapped to 0). Discard reviews with the rating 3 as neutral reviews. Include the number of reviews for each of these three classes in your report (to be printed by .py file in separate lines).

The original dataset is large. To avoid computational burden, select 100,000 reviews with positive sentiment along with 100,000 reviews with negative sentiment to preform the required tasks on the downsized dataset. Split your dataset into 80% training dataset and 20% testing dataset. 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)

Implement the following steps to preprocess the dataset you created:

- convert the all reviews into the lower case.
- remove the HTML and URLs from the reviews
- remove non-alphabetical characters

remove extra spaces

- perform contractions on the reviews, e.g., won't $\rightarrow$ will not. Include as many contractions in English that you can think of.

You can either use Pandas package functions or any other built-in functions. Do not try to implement the above processes manually. Most of the above processes can be performed with one line of code.

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

Print three sample reviews before and after data cleaning + preprocessing. In 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 which consists of features and binary labels for the reviews you selected.

5. Perceptron (10 points)

Train a Perceptron model on your training dataset using the sklearn built-in implementation. Report Accuracy, Precision, Recall, and f1-score on both the training and testing split of your dataset. These 8 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 Accuracy, Precision, Recall, and f1-score on both the training and testing split of your dataset. These 8 values should be printed in separate lines by the .py file.

Logistic Regression (10 points)

Train a Logistic Regression model on your training dataset using the sklearn built-in implementation. Report Accuracy, Precision, Recall, and f1-score on both the training and testing split of your dataset. These 8 values should be printed in separate lines by the .py file.

7. Multinomial Naive Bayes (10 points)

Train a Multinomial Naive Bayes model on your training dataset using the sklearn built-in implementation. Report Accuracy, Precision, Recall, and f1-score on both the training and testing split of your dataset. These 8 values should be printed in separate lines by the .py file.

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

- Statistics of three classes (with comma between them)
- Average length of reviews before and after data cleaning
- Average length of reviews before and after data preprocessing
- Accuracy, Precision, Recall, and f1-score for training and testing split (in the mentioned order) for Perceptron
- Accuracy, Precision, Recall, and f1-score for training and testing split (in the mentioned order) for SVM
- Accuracy, Precision, Recall, and f1-score for training and testing split (in the mentioned order) for Logistic Regression
- Accuracy, Precision, Recall, and f1-score for training and testing split (in the mentioned order) for Naive Bayes