CM30320 Natural Language Processing Assignment

Due 5th December 2023

Due:	5th December 2023 [Week 10, Thursday]
% of Module:	30
Marked out of:	Marks: 100
Submit (Where):	Moodle
Submit (What):	2 independent files (as detailed below)

This assignment has 5 pages

Learning Objectives

This assignment will give you the opportunity to experiment with sentiment analysis, an important task used in different aspects of natural language processing and text mining. In addition you will have the opportunity to experiment with ChatGPT. By the end of this assignment, you should be able to:

1. Text Classification: Develop the ability to classify text into different sentiment categories (e.g., positive, negative, neutral) using machine learning and natural language processing techniques.
2. Feature Extraction: Acquire the skills to identify and extract relevant features from text data that can be used to assess sentiment. This includes understanding the importance of features such as words, phrases, and context.
3. Model Selection and Evaluation: Explore various sentiment analysis models and techniques, and learn how to select the most suitable model for a given task. Evaluate model performance using appropriate metrics.
4. Understand the capabilities and limitations of pre-trained language models, specifically large language models. The rapid adoption of LLMs implies that you will more than likely be using these models in your everyday life, and this provides you with the opportunity to understand what they can do.

Submission Requirements:

You are required to upload two separate files to Moodle. Do NOT upload a single zip file.

1. A Report describing your methods (detailed below)
    a. This should be a PDF/Word file - Name this file “Report”
2. A zip file containing the program code associated with this part.
    a. This should be a zip file. Name this file “Code.zip”

Sentiment Analysis

30% of Module

This mini-project is based on the material covered in lectures and the programming exercises that you are provided with during this course. The application that you will be addressing is Sentiment Analysis, which is concerned with the automated identification of the opinion polarity associated with a particular piece of text. Most frequently, this is defined as identifying whether a particular piece of writing (e.g., a review on a movie or a product) is positive or negative, and this is precisely what you will be doing in this project. Specifically, you are provided with a set of reviews extracted from the Internet Movie Database (IMDb) with various polarity labels, and your task is to develop an application that can detect the sentiment polarity given any input text. This is a real-world application that is popular in an academic as well as industrial context.

Dataset

You will be using a subset of 2,000 positive and 2,000 negative movie reviews extracted from the Large Movie Review Dataset (https://ai.stanford.edu/~amaas/data/sentiment/). The reviews were extracted based on their star rating: reviews with a score <=4 stars are considered clearly negative, while those with a score >=7 are considered clearly positive. You are provided with a set of positive reviews (in the pos/ folder) as well as negative reviews (in the neg/ folder), where you can see that the file naming conventions are as follows: id_star.txt. In other words, if you'd like to benefit from the star rating assigned to a particular review, you can extract this information from the names of the files.

For background information on the task and the datasets, take a look at:

- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).
- The dataset webpage: https://ai.stanford.edu/~amaas/data/sentiment/

Assignment requirements

You will be marked on a combination of your report and the program code that you submit. Overall, your report should be self-contained and should include all the information required to grade your assignment. Similarly, your program code should be self-contained and should be well documented so it can be run if required.
The following are the tasks required in this part of your assignment. You must include these sections in your report.

Assignment Requirements and Marking Details

Marked out of: 100 (Breakdown below)
Programing objectives: See below
Requirements:
You must (at a minimum) explore the following methods. You must describe each of these in your report and write the associated Python scripts. You should provide enough detail in the report to ensure that someone who has not sat through the lectures can understand what you intend to do. Your code must be clearly documented.

<u>0. Clear Report that includes the following sections:</u>
Total Marks: 10

1. Introduction and summary of what worked for you (see below)
2. Description of experiments (see requirements below)
3. Clear experimental results driving your choice of features
4. **Be sure to include the following additional sections in your report.**
5. A quick summary of your results on the test set and observations on what worked and what did not work.

<u>1. Evaluation Data splits</u>
Total Marks: 5
Requirements: Split the data into three data splits: Train, Evaluation and Test. Describe what you did. No program code is required in your report.

<u>2. Feature Generation, Dimensionality reduction and Features Selection</u>
Total Marks: 50

<u>2.1 Feature Generation</u>: Generate features using the following methods.

1. [5 marks] [No code in report] Three different methods of tokenization, for example:
    a. Split by whitespace, include all tokens
    b. Lemmatized
    c. Stemmed.
2. [5 marks] [No code in report] Using all the tokens is likely to be impractical (the input to your classifier might be too large). You must drop some of the tokens you generated using some frequency distributions. Choose this cut-off based on downstream performance.

3. [10 marks] [No code in report] At least two methods of extracting compositional phrases, for example:
    a. Frequently occurring n-grams (where n is 2, 3, … found through experiments)
    b. PoS + constituency parsing for noun phrases,
4. [10 marks] [*Include* code in report] Any ONE method of either adding special features or boosting features counts (examples below)

    NOTE: This section is hard. You might want to attempt it only after you have completed the rest of the assignment. Write your code so your experiments can quickly be run with this variation included

    Example/Hint:

    You might want to "boost" (by some small number/fraction) the "count" of all hypernyms (or some other WordNet relation) of a word that occurs in the example.

    You might want to use WordNet or Wikipedia to identify phrases - notice that this is different from using n-grams as it brings in external knowledge.

2.2 Normalise: (10 marks) [Include code in report] Any TWO ways of normalising your word/token counts that MUST INCLUDE TF-IDF. Clearly describe your reasoning behind choosing the specific methods you chose to use. Include a well documented function to perform the normalisation in your report.

2.3 Features Selection (10 Marks): [No code in report] You will have to run a variety of experiments to convince the reader that the specific combination of features you have selected are the most effective of those you test. To this end you must:

1. Clearly describe the features you have chosen
2. Train the Naïve Bayes implementation from https://scikit-learn.org/stable/ using this subset of features.
3. Test the model using different combinations of features on the development set to demonstrate the best combination.
4. Present a table of performance so you can justify your choice.

At the end of this process you should have selected a "feature set" - evaluate the test set using this final feature set.

3. Naïve Bayes:

Marks: 25 marks (Must include program code in report)

Implement Naïve Bayes from scratch (clf = MultinomialNB is NOT acceptable). Do NOT use any existing implementations. You must document your code. You must include the complete (relevant) code in your report including relevant comments to demonstrate your understanding. Evaluate your implementation against that by Scikit learn that you previously used. You are to perform the evaluation on the test set. Any hyperparameter optimization must be done on the development set.

4. BERT

Marks: 10
This part of your assignment is completely independent of all previous sections. Using the same train/dev/test splits as before, use BERT, specifically the base model, to perform the same classification. Be sure to experiment with both the cased and uncased versions of the model.

You are only required to use the existing script for fine-tuning:
https://github.com/huggingface/transformers/tree/main/examples/pytorch/text-classification#text-classification