COM3110: Text Processing (2023/2024)

Assignment: Sentiment Analysis of Movie Reviews

1 Project Description

The aim of this project is to implement a multinomial Naive Bayes model for a sentiment analysis task using the Rotten Tomatoes movie review dataset. This dataset is derived from the "Sentiment Analysis on Movie Reviews" Kaggle competition[1], that uses data from the works of Pang and Lee [1] and Socher at al. [2]. Obstacles like sentence negation, sarcasm, terseness, language ambiguity, and many others make this task very challenging.

2 Submission

Submit your assignment work electronically via Blackboard. Precise instructions for what files to submit are given later in this document. Please check you have access to the relevant Blackboard unit and contact the module lecturer if not.

SUBMISSION DEADLINE: 15:00, Friday week 12 (15th December, 2023)

Penalties: standard departmental penalties apply for late hand-in and use of unfair means.
Important: the use of any generative AI tools in the preparation of the solution to this work (both code and report) is not permitted.

3 Materials Provided

Download the file `COM3110_assignment2_files.zip` from the Blackboard course. It unzips to a folder that contains a number of code and data files, for use in the assignment.

`NB_sentiment_analyser.py` is the starter code for this project. It contains the standard input and output requirements. You should use this file as you `main` method and should ensure that any additional output (that you may have added to help debug your program) is removed before submission (see more details below).

3.1 Dataset: Movie Reviews

The dataset in the `moviereviews` folder is a corpus of movie reviews originally collected by Pang and Lee [1]. This dataset contains tab-separated files with phrases from the Rotten Tomatoes dataset. The data are split into **train/dev/test** sets and the sentences are shuffled from their original order.

- Each sentence has a **SentenceId**.
- Sentences have already been **tokenized**.

[1] `https://www.kaggle.com/competitions/sentiment-analysis-on-movie-reviews/overview`

The training, dev and test set contain respectively 6529, 1000 and 1000 sentences. The sentences are labelled on a scale of five values:

0. negative
1. somewhat negative
2. neutral
3. somewhat positive
4. positive

In the following table you can find several sentences and their **sentiment score**. Please note that the **test set is "blind"**, i.e. you are not given the gold standard sentiment scores. You will need to submit some files with the predicted labels for the **test set** and we will use these as part of your assessment (see below).

SentenceId	Sentence	Sentiment
1292	The Sweetest Thing leaves a bitter taste .	0
343	It labours as storytelling	1
999	There 's plenty to enjoy – in no small part thanks to Lau .	3
1227	Compellingly watchable .	4

4 Evaluation

Systems are evaluated according to:

- **macro-$F1$ score**, i.e. the mean of the class-wise $F1$-scores:

$$\text{macro–}F1 = \frac{1}{N}\sum_{i=0}^{N} F1\text{–score}_i$$

where N is the number of classes. $F1$–score is calculated for each class i:

$$F1\text{–score}_i = \frac{2 * \text{Precision}_i * \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} = \frac{2 * TP_i}{2 * TP_i + FP_i + FN_i}$$

TP = true positives
TN = true negatives
FP = false positives
FN = false negatives

5 Project Roadmap

5.1 Implementation

1. Preprocessing steps:
   - You are free to add any preprocessing step (e.g. lowercasing) before training your models. Explain what you did in your report. Please note that these preprocessing steps are "universal", i.e. they should be applied for all models you train.
   - Implement a function to map the 5-value sentiment scale to a 3-value sentiment scale. Namely, the labels "negative" (value 0) and "somewhat negative" (value 1) are merged into label "negative" (value 0). "Neutral" (value 2) will be mapped to "neutral" (value 1). And finally, "somewhat positive" (value 3) and "positive" (value 4) will be mapped to the label "positive" (value 2).
2. Implement a **multinomial Naive Bayes classifier with Laplace smoothing from scratch**.
   - You may **NOT** re-use already implemented classes/functions (e.g. scikit-learn)
3. For each set of labels (5-value and 3-value scales), train **two different models** (i.e. 4 models in total):
   (a) One considering all the words in the training set as features (after your defined preprocessing steps).
   (b) One with a set of **features of your choice**, determined by your experience or empirical experiments (you will explain how you selected the features in your short report). These features can be selected with the help of your development data, i.e. you can use the development set to explore different feature combinations and submit the results for your best model.
   The selected features need to be complex and well-defined, with a solid hypothesis on why you expect them to perform well, based on the reading material provided in class or other literature on sentiment analysis. For instance, a simple selection of "adjectives as features" will not achieve a high mark in this part, unless you performed a rather exhaustive benchmarking considering different features and concluded that the simple approach is the best.
4. Implement the macro-$F1$ score from scratch.
5. Compute and display confusion matrices.
6. Process the test data with your best performing models (one for each class setting). See details below on how to submit these results

5.2 Report

You will submit a short report **NOT EXCEEDING 3 PAGES IN LENGTH** (appendices are not allowed), briefly linking specific sections to your code, explaining the feature selection process, presenting and analysing the results (using tables and graphs) and performing an error analysis. Graphs/tables should be used in presenting your results, to aid exposition.

Your report must have the following sections:

1. **Implementation details**
   In this section you should add reference to the files and lines of your code where the following aspects are implemented:

   - Prior calculation.
   - Likelihoods calculation.
   - Feature extraction.
   - macro-$F1$ score.

   You should also explain your **pre-processing steps** and any practices you used to improve the performance of your code.

2. **Feature selection**
   In this section you will explain your choice for features. You will need to explain your intuition (hypothesis) behind the chosen features, highlighting their potential benefits as well as limitations. You should also explain which tools, models, approaches you used for extracting your features. As mentioned in the previous section, a simple use of "adjectives as features" without any justification will not result in high marks. Ideally, you will either find evidence in the literature (e.g. the reading material from the lectures or research papers in the area of sentiment analysis / text classification) or you will perform an extensive/exhaustive exploration of features in order to make your selection.

3. **Results and Discussion**
   In this section you will summarise the results for the three different models you trained (all words and your selected features) in the two different settings (5-value and 3-value scales). You should use tables and graphs to present your results, highlighting the best models according to macro-$F1$ and confusion matrices. Here you should explicitly present which is your best model and explain why.

4. **Error analysis**
   For this section, you will select four examples from the development set, showcasing the weaknesses of your best model for each setting (i.e. 3-class or 5-class). Based on this analysis, you should discuss why the model fails for these examples and hypothesise what can be done to improve the model. The selected four examples will also be assessed, as they should represent different challenging cases for your model.

5. **References**
   Citations to books, journals and conference papers that you used to develop your model or to support your argumentation in the report. References can use an extra page that is not counted in the page limit. You should IEEE referencing style.[2]

6 What to Submit

Your assignment work is to be submitted electronically using Blackboard. You will submit a **ZIP file** named

`studentID_Assignment2.zip`

where **studentID is the ID that you use to login into MUSE** (i.e. the IDs starting with "acp", "mm", etc).

[2] `https://librarydevelopment.group.shef.ac.uk/referencing/ieee.html`

This ZIP file should include:

1. **Your Python code**.
   You should use Python 3.9.x (or above)[3] and consider the provided `NB_sentiment_analyser.py` file as your main file. You can (and you are encouraged to) create other files to organise your code in classes (therefore, the final submission is composed of all the files needed to run your code). However, the "interface" of your project should remain the `NB_sentiment_analyser.py` file. To run and test your code, we will use the already pre-defined parameters in a command line:

   ```
   python NB_sentiment_analyser.py <TRAINING_FILE> <DEV_FILE> <TEST_FILE> -classes
   <NUMBER_CLASSES> -features <all_words,features> -output_files -confusion_matrix
   ```

   where:

   - `<TRAINING_FILE> <DEV_FILE> <TEST_FILE>` are the paths to the training, dev and test files, respectively;
   - `-classes <NUMBER_CLASSES>` should be either 3 or 5, i.e. the number of classes being predicted;
   - `-features` is a parameter to define whether you are using your selected features or no features (i.e. all words);
   - `-output_files` is an optional value defining whether or not the prediction files should be saved (see below – default is "files are not saved"); and
   - `-confusion_matrix` is an optional value defining whether confusion matrices should be shown (default is "confusion matrices are not shown").

   A **standard output** of your program is also already pre-defined and available in the `NB_sentiment_analyser.py` file. It is a tab-separated output that will contain:

   ```
   studentID[tab]Number of classes[tab]Features[tab]macro-F1(dev)
   ```

   For instance, for the following input:

   ```
   python NB_sentiment_analyser.py train.tsv dev.tsv test.tsv -classes 3 -features
   all_words
   ```

   where we want the results for **3 classes** and using **all words** as features, the expected program output is:

   ```
   acpXXjd[tab]3[tab]all_words[tab]0.200
   ```

   Ensure that any additional output (that you may have added to help debug your program) is **removed before submission**. In other words, the final submission should only print the above line with your results. In order to add your studentID to the code, you need modify line 12 of the `NB_sentiment_analyser.py` file.

[3]Please be mindful that the latest python release (3.12) may not be very stable yet.

2. A **README file** containing all the details about your implementation that are needed to run your code. You are expected to use Python 3.9.x (or above), however, you have any compelling reason to use a different version you should clearly explain in this README. In this file you should also include all the libraries that you used. Standard libraries like numpy and pandas do not required much detail (unless you rely on a specific version). However, if you use any other non-standard library (e.g. when extracting features for the Naive Bayes model), you need to detail their installation here.

   Recommendation: The use of a `requirements.txt` file to install[4] the libraries needed for your project is strongly recommended.

3. **Four files** with the **predictions** on the **development and test** sets, using your **best model** (either with or without your features) in each setting, i.e. either 3 or 5 classes.
   The format is tab separated as follows : **SentenceId[tab]Sentiment**
   An example file named "SampleSubmission_test_predictions_5classes_acpXXjd.tsv" is provided with the data.
   Those files **MUST BE NAMED** respectively:

   - `dev_predictions_3classes_studentID.tsv`
   - `test_predictions_3classes_studentID.tsv`
   - `dev_predictions_5classes_studentID.tsv`
   - `test_predictions_5classes_studentID.tsv`

   where **studentID is the ID that you use to login into MUSE (i.e. the IDs starting with "acp", "mm", etc)**.
   We will use these files to calculate the performance of your best system on the development set and test set.

4. The **short report (as a PDF file)** (as described in the previous section).

7 Assessment Criteria

A total of 25 marks are available for the assignment and will be assigned based on the following general criteria (a more detailed marking codebook will be released later).

Implementation and Code Style – including README file (10 marks)
Have appropriate Python constructs been used? Is the code comprehensible and clearly commented? Does your code run and follow the pre-defined instructions? Is the Naive Bayes implementation correct? Were all the functionalities adequately implemented? Does the code run in an appropriate time (i.e. does not exceed 3 minutes)?

Report (15 marks)
Is the report a clear and accurate description of the implementation? Have you explained in detail the hypothesis and/or empirical experimentation performed for feature selection? How complete and accurate is the discussion of the performance of the different systems under a range of configurations? How do you choose which is the best model? Did your models show improvements over a majority class baseline? Did you provide a comprehensive error analysis?

[4] See more at `https://pip.pypa.io/en/stable/reference/requirements-file-format/`

8 Notes and Comments

- Consider using the **Pandas** library to load the data `https://pandas.pydata.org/`.
- Consider using **Seaborn heatmap** to render the confusion matrices `https://seaborn.pydata.org/`.
- You may search internet for lists of English punctuation and/or stopwords (also called function words) that you may use in your assignment.
- **sklearn** functions (such as CountVectorized) *SHOULD NOT BE* used. All Naive Bayes calculations (including the count of words) should be made from scratch (you can use **numpy**).
- For your information, the majority class macro-$F1$ results in the dev set for the different class settings are:
  - 3-class: 0.200
  - 5-class: 0.089

References

[1] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 115–124. [Online]. Available: https://aclanthology.org/P05-1015

[2] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://aclanthology.org/D13-1170