# COM3110 : Text Processing (2022/2023)

## Assignment: Sentiment Analysis of Movie Reviews

## 1 Project Description

The aim of this project is to implement a corpus-based 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, 2005] and [Socher et al., 2013]. 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 11 (9th December, 2022)**

Penalties: standard departmental penalties apply for late hand-in and use of unfair means

## 3 Data Description

The dataset is a corpus of movie reviews originally collected by Pang and Lee. 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**.
- They all have been **tokenized** already.

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

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

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 to as part of your assessment (see below).

| SentenceId | Phrase | 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 **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 $i$:

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

## 5 Project Roadmap

1. Implement 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 this 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 Naive Bayes classifier **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):
   - One considering all the words in the training set as features (after your defined preprocessing steps).
   - One with a set of **features** of your choice determined by your experience (you will explain how you selected the features in your short report).
4. Implement the macro-$F1$ score metric from scratch.

5. Compute and display confusion matrices on the development set for each developed models. Compare the results using confusion matrices and macro-$F1$.

6. Process the test data with your best performing models (one for each class).

7. Write a report (see below for details).

## 6 What to Submit

Your assignment work is to be submitted electronically using MOLE, and should include:

1. **Your Python code**.
   You should use Python 3.9.x (or above)[2] 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 be through this provided 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:

   ```
   Student [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
   ```

[2]Please be mindful that the latest python release (3.11) may not be very stable yet.

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

```
acpXXjd [tab] 3 [tab] False [tab] 0.200
```

2. A **README file** containing all the details about your implementation that are needed to run your code. You are expected to use Python 3.10, 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 are not required much details (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.

3. **Four files with the predictions on the development and test corpora considering 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_<USER_ID>.tsv
   - test_predictions_3classes_<USER_ID>.tsv
   - dev_predictions_5classes_<USER_ID>.tsv
   - test_predictions_5classes_<USER_ID>.tsv

   where **USER_ID is the student 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. A **short report (as a pdf file)**.
It should **NOT EXCEED 2 PAGES IN LENGTH**. The report should include a brief description of the extent of the implementation achieved, and should present the performance results you have collected under different configurations, and any conclusions you draw from your analysis of these results. Graphs/tables may be used in presenting your results, to aid exposition.

## 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 (15 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?

**Report (10 marks)**
Is the report a clear and accurate description of the implementation? 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?

## 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

[Pang and Lee, 2005] Pang, B. and Lee, L. (2005). 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)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

[Socher et al., 2013] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.