COMP9414: Artificial Intelligence
Assignment 2: Sentiment Analysis

Due Date: Week 9, Friday, July 31, 11:59 p.m.

Value: 25%

This assignment is inspired by a typical real-life scenario. Imagine you have been hired as a Data Scientist by a major airline company. Your job is to analyse the Twitter feed to determine customer sentiment towards your company and its competitors.

In this assignment, you will be given a collection of tweets about US airlines. The tweets have been manually labelled for *sentiment*. Sentiment is categorized as either *positive*, *negative* or *neutral*. **Important: Do not distribute these tweets on the Internet, as this breaches Twitter's Terms of Service.**

You are expected to assess various supervised machine learning methods using a variety of features and settings to determine what methods work best for sentiment classification in this domain. The assignment has two components: *programming* to produce a collection of models for sentiment analysis, and a *report* to evaluate the effectiveness of the models. The programming part involves development of Python code for data preprocessing of tweets and experimentation of methods using NLP and machine learning toolkits. The report involves evaluating and comparing the models using various metrics, and comparison of the machine learning models to a baseline method.

You will use the NLTK toolkit for basic language preprocessing, and scikit-learn for feature construction and evaluating the machine learning models. You will be given an example of how to use NLTK and scikit-learn for this assignment (`example.py`). For the sentiment analysis baseline, NLTK includes a hand-crafted (crowdsourced) sentiment analyser, VADER,[1] which may perform well in this domain because of the way it uses emojis and other features of social media text to intensify sentiment, however the accuracy of VADER is difficult to anticipate because: (i) crowdsourcing is in general highly unreliable, and (ii) this dataset might not include much use of emojis and other markers of sentiment.

Data and Methods

A *training* dataset is a tsv (tab separated values) file containing a number of tweets, with one tweet per line, and linebreaks within tweets removed. Each line of the tsv file has three fields: instance_number, tweet_text and sentiment (positive, negative or neutral). A *test* dataset is a tsv file in the same format as the training dataset *except that your code should ignore the sentiment field*. Training and test datasets can be drawn from a supplied file `dataset.tsv` (see below).

For all models except VADER, consider a tweet to be a collection of words, where a *word* is a string of at least two letters, numbers or the symbols #, @, _, $ or %, delimited by a space, after removing all other characters (two characters is the default minimum word length for CountVectorizer in scikit-learn). URLs should be treated as a space, so delimit words. Note that deleting "junk" characters may create longer words that were previously separated by those characters.

Use the supervised learning methods discussed in the lectures: Decision Trees (DT), Bernoulli Naive Bayes (BNB) and Multinomial Naive Bayes (MNB). Do not code these methods: instead use

[1] https://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109

the implementations from scikit-learn. Read the scikit-learn documentation on Decision Trees[2] and Naive Bayes,[3] and the linked pages describing the parameters of the methods. Look at `example.py` to see how to use CountVectorizer and train and test the machine learning algorithms, including how to generate metrics for the models developed.

The *programming* part of the assignment is to produce DT, BNB and MNB models and your own model for sentiment analysis in Python programs that can be called from the command line to train and classify tweets read from correctly formatted tsv files. The *report* part of the assignment is to analyse these models using a variety of parameters, preprocessing tools, scenarios and baselines.

Programming

You will produce and submit **four** Python programs: (i) `DT_sentiment.py` (ii) `BNB_sentiment.py`, (iii) `MNB_sentiment.py` and (iv) `sentiment.py`. The first three of these are standard models as defined below. The last is a model that you develop following experimentation with the data. Use the given dataset (`dataset.tsv`) containing 5000 labelled tweets to develop the models.

These programs, when called from the command line with two file names as arguments, the first a training dataset and the second a test dataset, should print (to standard output), the instance_number and sentiment produced by the classifier of each tweet in the test set when trained on the training set (one per line with a space between them) – each sentiment being the string "positive", "negative" or "neutral". For example:

```
python3 DT_sentiment.py training.tsv test.tsv > output.txt
```

should write to the file `output.txt` the instance number and sentiment of each tweet in `test.tsv`, as determined by the Decision Tree classifier trained on `training.tsv`.

When reading in training and test datasets, make sure your code reads *all* the instances (some Python readers use "excel" format, which uses double quotes as separators).

Standard Models

Train the three standard models on the supplied dataset of 5000 tweets (the whole of `dataset.tsv`). For Decision Trees, use scikit-learn's Decision Tree method with criterion set to 'entropy' and with random_state=0. Scikit-learn's Decision Tree method does not implement pruning, rather you should make sure Decision Tree construction stops when a node covers fewer than 50 examples (1% of the training set). Decision Trees are likely to lead to fragmentation, so to avoid overfitting and reduce computation time, for all Decision Tree models use as features only the 1000 most frequent words from the vocabulary (after preprocessing to remove "junk" characters as described above). Write code to train and test a Decision Tree model in `DT_sentiment.py`.

For both BNB and MNB, use scikit-learn's implementations, but use *all* of the words in the vocabulary as features. Write two Pythons programs for training and testing Naive Bayes models, one a BNB model and one an MNB model, in `BNB_sentiment.py` and `MNB_sentiment.py`.

Your Model

Develop your best model for sentiment classification by varying the number and type of input features for the learners, the parameters of the learners, and the training/test set split, as described in your report (see below). Submit one program, `sentiment.py`, that trains and tests a model.

[2] https://scikit-learn.org/stable/modules/tree.html

[3] https://scikit-learn.org/stable/modules/naive_bayes.html

Report

In the report, you will first evaluate the standard models, then present your own model. For evaluating all models, report the results of training on the first 4000 tweets in `dataset.tsv` (the "training set") and testing on the remaining 1000 tweets (the "test set"), rather than using the full dataset of 5000 tweets for training, so stopping the Decision Tree classifiers when nodes cover less than 40 tweets rather than 50. Use the metrics (micro- and macro-accuracy, precision, recall and F1) and classification reports from scikit-learn. Show the results in either tables or plots, and write a *short* paragraph in your response to each item below. The answer to each question should be self contained. **Your report should be at most 10 pages.** Do not include appendices.

1. (1 mark) Give simple descriptive statistics showing the frequency distribution for the sentiment classes for the whole dataset of 5000 tweets. What do you notice about the distribution?

2. (2 marks) Develop BNB and MNB models from the training set using (a) the whole vocabulary, and (b) the most frequent 1000 words from the vocabulary (as defined using CountVectorizer, after preprocessing by removing "junk" characters). Show all metrics on the test set comparing the two approaches for each method. Explain any similarities and differences in results.

3. (2 marks) Evaluate the three standard models with respect to the VADER baseline. Show all metrics on the test set and comment on the performance of the baseline and of the models relative to the baseline.

4. (2 marks) Evaluate the effect of preprocessing the input features by applying NLTK English stop word removal then NLTK Porter stemming on classifier performance for the three standard models. Show all metrics with and without preprocessing on the test set and explain the results.

5. (2 marks) Evaluate the effect that converting all letters to lower case has on classifier performance for the three standard models. Show all metrics with and without conversion to lower case on the test set and explain the results.

6. (6 marks) Describe your best method for sentiment analysis and justify your decision. Give some experimental results for your method trained on the training set of 4000 tweets and tested on the test set of 1000 tweets. Provide a brief comparison of your model to the standard models and the baseline (use the results from the previous questions).

Submission

- Submit all your files using a command such as (this includes Python code and report):

```
give cs9414 ass2 DT*.py BNB*.py MNB*.py sentiment.py report.pdf
```

- Your submission should include:
  - Your `.py` files for the specified models and your model, plus any `.py` "helper" files
  - A `.pdf` file containing your report
- When your files are submitted, a test will be done to ensure that one of your Python files runs on the CSE machine **(take note of any error messages printed out)**
- When running your code on CSE machines:
  - Set `SKLEARN_SITE_JOBLIB=TRUE` to avoid warning messages
  - Do not download NLTK in your program: CSE machines have NLTK installed
- Check that your submission has been received using the command:

```
9414 classrun -check ass2
```

Assessment

Marks for this assignment are allocated as follows:

- Programming (auto-marked): 10 marks
- Report: 15 marks

Late penalty: 5 marks per day or part-day late off the mark obtainable for up to 3 (calendar) days after the due date.

Assessment Criteria

- Correctness: Assessed on standard input tests, using calls such as:

  ```
  python3 DT_sentiment.py training.tsv test.tsv > output.txt
  ```

  Each such test will give two files, a training dataset and a test dataset, which contain **any** number of tweets (one on each line) in the correct format. The training and test datasets can have any names, not just `training.tsv` and `test.tsv`, so read the file names from `sys.argv`. The output should be a sequence of lines (one line for each tweet) giving the instance number and classified sentiment, separated by a space and with no extra spaces or lines. There are 2 marks allocated for correctness of each of the three standard models.

  For your own method, 4 marks are allocated for correctness of your methods on test sets of tweets that include unseen examples.

- Report: Assessed on correctness and thoroughness of experimental analysis, and clarity and succinctness of explanations.

  There are 9 marks allocated to items 1–5 as above, and 6 marks for item 6. Of these 6 marks, 2 marks are for the explanation of your choice of model, 2 marks are for the experimental analysis of your model, and 2 marks are for the evaluation of your model in comparison to the standard models and baseline.

Plagiarism

Remember that ALL work submitted for this assignment must be your own work and no code sharing or copying is allowed. You may use code from the Internet only with suitable attribution of the source in your program. *Do not use public code repositories.* All submitted assignments will be run through plagiarism detection software to detect similarities to other submissions, including from past years. You should **carefully** read the UNSW policy on academic integrity and plagiarism (linked from the course web page), noting, in particular, that *collusion* (working together on an assignment, or sharing parts of assignment solutions) is a form of plagiarism. There is also a new plagiarism policy starting this term with more severe penalties.