COMP9414: Artificial Intelligence
Assignment 2: Opinion Mining

Due Date: Week 10, Friday, August 9, 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 political party around the time of the federal election. Your job is to analyse the Twitter feed concerning federal politics, to work out voter sentiment and the topics they are most concerned about.

In this assignment, you will be given a collection of tweets from the 2016 Australian Federal Election, a subset of those posted during the election campaign that contained the `#auspol` hashtag. The tweets have been labelled by domain experts for *sentiment* and *topic* (and also the *entity* the tweet is targeted towards, but that is not part of this assignment). **Important: Do not distribute these tweets on the Internet, as this breaches Twitter's Terms of Service.**

Sentiment is categorized as either *positive*, *negative* or *neutral*. Topics are assigned from a predefined list of 20 topics determined from several independent sources, and include categories such as economic management, refugees, healthcare, education, etc. Every tweet in your dataset is assigned exactly one topic, but in general not all election tweets have a topic (e.g. some tweets express an opinion about an entity without being about a clear topic), and some tweets have more than one topic. In addition, some tweets are labelled as *sarcastic*, where the meaning is different from the normal meaning of the words (often the opposite meaning). It is well known that sarcasm detection, and sentiment analysis of sarcastic tweets, are difficult problems.

You are expected to assess various supervised machine learning methods using a variety of features to determine what methods work best for a variety of tasks relating to sentiment and topic classification, on a variety of subsets of the data (e.g. the dataset with and without sarcastic tweets). Thus the assignment will involve some development of Python code for data preprocessing, and experimentation using machine learning toolkits. You will compare your models to reasonable baselines. The assignment has two components: *programming* to produce a collection of models for sentiment analysis and topic classification, and a *report* on the effectiveness of the models.

You will use the NLTK toolkit for basic language preprocessing, and scikit-learn for evaluating the machine learning models. The reason for this is that NLTK is comparatively poorly documented and maintained (its code is not coherently designed or well integrated, but it does include some useful preprocessing tools),[1] whereas scikit-learn is well maintained and easier to use, and includes standard implementations of machine learning algorithms. You will be given examples of how to use scikit-learn for this assignment.

For a sentiment analysis baseline, NLTK includes a hand-crafted (crowdsourced) sentiment analyser, VADER,[2] 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. Hence for both sentiment analysis and topic classification, use the majority class classifier as a baseline.

[1] NLTK's Decision Tree algorithm does not split on entropy, rather entropy is used for pruning (along with tree depth and support). NLTK has a standard implementation of Bernoulli Naive Bayes, but no implementation of Multinomial Naive Bayes. Methods to calculate metrics in different modules use inconsistent data formats, etc.

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

The *full* dataset is a tsv (tab separated values) file containing 2000 tweets, with one tweet per line. Each tweet is classified into sentiment (positive, negative or neutral) and *exactly one* topic. Linebreaks within tweets have been removed. Each line of the tsv file contains the following entries: instance_number, tweet_text, topic_id, sentiment, and is_sarcastic. A mapping from ids to topic_names is at the end of this document. For all models, consider a tweet to be a collection of words, where a *word* is a string of letters, numbers, or symbols ', #, @, $ or _ delimited by a space. Any other characters (and in particular URLs) should be treated as a space.

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 the implementations from scikit-learn. Read the scikit-learn documentaton on Decision Trees[3] and Naive Bayes,[4] and the linked pages describing the parameters of the methods.

The *programming* part of the assignment is to produce DT, BNB and MNB models (specified below) and two of your models for sentiment analysis and topic classification, in Python programs that can be called from the command line to classify tweets read from a file. The *report* part of the assignment is to analyse these models using a variety of parameter settings, preprocessing tools, scenarios and baselines.

Programming

You will produce and submit **eight** Python programs: (i) `DT_sentiment.py` (ii) `DT_topics.py`, (iii) `BNB_sentiment.py`, (iv) `BNB_topics.py`, (v) `MNB_sentiment.py`, (vi) `MNB_topics.py`, (vii) `sentiment.py` and (viii) `topics.py`. The first six of these are standard models as described below. The last two are models that you develop following experimentation with the data.

These programs, when called with a file name as argument from the command line, given a file that contains tweets (one per line) should output (to standard output), the classification of the tweet (one per line) – either a sentiment (positive, negative or neutral) or a topic_id. For example, the command:

```
python3 DT_sentiment.py testfile1.txt > output1.txt
```

should write the sentiment returned by the Decision Tree sentiment classifier for each tweet in `testfile1.txt` to the file `output1.txt`.

Standard Models

Train the six standard models on the full dataset. For Decision Trees, use scikit-learn's Decision Tree method with criterion set to 'entropy' and with random_state=0. Scikit-learn's DT method does not implement pruning, rather you should make sure Decision Tree construction stops when a node covers 1% (20) or fewer examples. Decision Trees are likely to lead to fragmentation, so to avoid overfitting and reduce computation time, use as features only the 200 most frequent words from the vocabulary. Produce two Decision Tree models: `DT_sentiment.py` and `DT_topics.py`.

For both BNB and MNB, use scikit-learn's implementations, but use *all* of the words in the vocabulary as features. Produce two BNB and two MNB models: `BNB_sentiment.py` and `BNB_topics.py`, and `MNB_sentiment.py` and `MNB_topics.py`.

Your Models

Develop your best models for sentiment and topic classification by varying the number and type of input features for the learners, the parameters of the learners, the training/test set split, as described in your report (see below). Submit two programs: `sentiment.py` and `topics.py`.

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

Report

In the report, you will first summarize and comment on the standard methods, then show the results of experiments used to develop your own models (one model for sentiment analysis and one for topic classification). For evaluating methods, report the results of training a model on the first 1500 tweets in the dataset (the *training* set) and testing on the remaining 500 tweets (the *test* set), rather than using the full dataset of 2000 tweets. Show the results of statistics or metrics as either a table or a plot for both training and test sets, and write a *small* paragraph in your response to each item below (though you may need more space to explain the development of your own methods). Use metrics calculated by scikit-learn, i.e. accuracy, micro and macro precision, recall and F1, and the classification report produced by scikit-learn.

1. (1 mark) Give simple descriptive statistics showing the frequency distributions for the sentiment and topic classes across the full dataset. What do you notice about the distributions?

2. (2 marks) Vary the number of words from the vocabulary used as training features for the methods (e.g. the top N features for $N = 100, 200$, etc.). Show metrics calculated on both the training set and the test set. Explain any difference in performance of the models between training and test set, and comment on metrics and runtimes in relation to the number of features.

3. (2 marks) Evaluate the standard models with respect to baseline predictors (VADER for sentiment analysis, majority class for topic classification). Comment on the performance of the baselines and of the methods relative to the baselines.

4. (2 marks) Evaluate the effect that preprocessing the input features, in particular stop word removal and Porter stemming as implemented in NLTK, has on classifier performance, for the three methods for both sentiment and topic classification. Compare results with and without preprocessing on training and test sets and comment on any similarities and differences.

5. (2 marks) Sentiment classification of neutral tweets is notoriously difficult. Repeat the experiments of items 3, 2 and 4 for sentiment analysis with only the positive and negative tweets (i.e. removing neutral tweets from both training and test sets). Compare these results to the previous results. Is there any difference in the metrics for either of the classes (i.e. consider positive and negative classes individually)?

6. (6 marks) Describe your best method for sentiment analysis and your best method for topic classification. Give some experimental results showing how you arrived at your methods. Now provide a similar evaluation of your best methods in relation to the standard methods and the baselines considering all the above issues.

Submission

- Submit all your files using the following command (this includes Python code and report):

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

- Your submission should include:
  - Your **eight** `.py` source files for the specified models and your models
  - 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)**
- 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: 3 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 testfile1.txt > output1.txt
  ```

  Each such test will give a file, in this example called `testfile1.txt`, which contains a number of labelled tweets (one on each line) in the same format as in the dataset; the output should be a sequence of lines (one line for each tweet) giving the classification (either a sentiment or topic_id), with no extra spaces or lines. There is 1 mark allocated for correctness of each of the six standard models.

  For your own methods, 4 marks are allocated for correctness of your methods on a test set of labelled tweets that includes unseen examples.

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

  There are 10 marks allocated to items 1–5 as above, and 6 marks for item 6. Of these 6 marks, 3 are for the sophistication of your models and 3 for your experimental analysis and evaluation of the methods.

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. All submitted assignments will be run through plagiarism detection software to detect similarities. 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.

Topics

ID	Topic
10000	corruption/governance
10001	employment/jobs
10002	tax/negative gearing
10003	economic management
10004	superannuation
10005	healthcare/medicare
10006	social issues/marriage equality/religion
10007	indigenous affairs
10008	asylum seekers/refugees
10009	early education and child care
10010	school education
10011	higher education
10012	innovation/science/research
10013	environment/climate change
10014	infrastructure
10015	telecommunications/nbn
10016	terrorism/national security
10017	foreign policy
10018	agriculture/irrigation/dairy industry
10019	mining and energy