University of Sheffield | Department of Computer Science

COM6115 Text Processing

Assessment: Sentiment Analysis

Changelog

Ver 1.1	16/11/2023	• Assessment brief:
		  ○ Rephrasing of Step 4.
		  ○ In Step 4, changed '50' into '100' most useful words, as is in Sentiment.py.
		  ○ Rephrasing of Step 6.1.
		• Sentiment.py: Added code for handling headers of lexicon files.

Ver 1.0	09/11/2023	Initial Release.

Quick Summary

To better understand the strengths and limitations of Bayesian text classification, in this assignment you are going to investigate Sentiment Analysis using two sentiment datasets you will be provided. You will also be provided a python script that implements Naive Bayes. You will need to write a report (**no more than 1000 words**) to describe your results and findings.

Note: This assessment accounts for 30% of your total mark for the course. Your report may be submitted for a plagiarism check (e.g., Turnitin). For any clarification on this assessment, please use the Discussion Board on Blackboard.

Assessment Tasks

STEP 1

Download the data from Blackboard. This contains the following:

1. A dataset with snippets of **movie reviews from the Rotten Tomatoes website** (one text file for positive reviews and one text file for negative reviews):
   a. `rt-polarity.pos`
   b. `rt-polarity.neg`
2. A smaller dataset with snippets of **reviews for Nokia phones** (again, 2 files):
   a. `nokia-pos.txt`
   b. `nokia-neg.txt`
3. A **sentiment dictionary** of positive and negative sentiment words:
   a. `negative-words.txt` contains 4783 negative-sentiment words
   b. `positive-words.txt` contains 2006 positive-sentiment words
4. A **python script** called `Sentiment.py` (you will need Python 3 to run it).
   This includes: an implementation of Naive Bayes, a knowledge-based classifier using the sentiment dictionary, and some helper functions.

STEP 2

Run Naive Bayes on Rotten Tomatoes Data:

1. The code splits the Rotten Tomatoes Data into a training and test set in `readFiles()`, then builds the p(word|sentiment) model on the training data in `trainBayes()`, and finally applies Naive Bayes to the test data in `testBayes()`.

2. Write a function which will print out **Accuracy**, **Precision**, **Recall** and **F-measure** for the test data.

[5 pt]

3. Run the code and report the classification results.

[5 pt]

STEP 3

Run Naive Bayes on Nokia Data:

1. In the python script, towards the end of the file (lines 272 and 274), uncomment out the other two calls to `testBayes()`. These run Naive Bayes on the training data and on Nokia product reviews.

2. What do you observe? Why are the results so different?

[10 pt]

STEP 4

What is being learnt by the model?

1. Which are the most useful words for predicting sentiment? The code you have downloaded contains another function `mostUseful()` that prints the most useful words for deciding sentiment. Uncomment the call to `mostUseful(pWordPos, pWordNeg, pWord, 100)` at the bottom of the program, and run the code again. This prints the words with the highest predictive value. Add these words (for both positive and negative sentiment) to your report. You can add them in an Appendix, in either text form or as a screenshot. (These do not count to the total word count).

   [5 pt]

2. Are the words selected by the model good sentiment terms? How many of them are in the sentiment dictionary?

   [5 pt]

STEP 5

How does a rule-based system compare to Naive Bayes?

1. Add some code for the function `testDictionary()` which will print out Accuracy, Precision, Recall and F-measure for the test data.

   [5 pt]

   Uncomment out the three lines towards the end of the program that call the function `testDictionary()` and run the program again. All this code does is add up the number of negative and positive words present in a review and predict the larger class.

2. How does the dictionary-based approach compare to Naive Bayes on the two domains? What conclusions do you draw about statistical and rule-based approaches from your observations?

   [5 pt]

3. Write a new function to improve the rule-based system, e.g., to take into account negation, diminisher rules, etc. Run the program again and analyse the results on both datasets.

   [25 pt]

STEP 6

Error Analysis:

1. Comment out all but one of the `testBayes` and `testDictionary` calls (one for each, two in total).

2. At the top of the program, set `PRINT_ERRORS=1`

3. Run the program again, and it will print out the mistakes made. List the mistakes in the report.

[5pt]

4. Please explain why the model is making mistakes (e.g., analyse the errors and report any patterns or generalisations).

[15 pt]

Marking Criteria

Along with your **code**, you should submit a **report** to describe your results and findings by following the tasks detailed in the 6 steps above.

Here is a summary of the marking criteria:

1. Quality of the report, including structure and clarity. No more than 1000 words. **[15 pt]**
2. Step 2 **[10 pt]**
3. Step 3 **[10 pt]**
4. Step 4 **[10 pt]**
5. Step 5 **[35 pt]**
6. Step 6 **[20 pt]**

Submission Guidelines

You should submit a **PDF version of your report along with your code** via Blackboard by **23:59, Friday 8th December 2023**.

The **name of the PDF file** should have the form:

`COM6115_Assessment-SA_<surname>_<first_name>_<student_ID>.pdf`

For instance, “COM6115_Assessment-SA_Smith_John_XXXXX.pdf”, where XXXXX is your student ID (i.e. your username for logging into Blackboard).