CSCE-638, Programming Assignment #4 Sentiment Lexicon Induction
Due: Monday, Dec. 4, 2022 by 11:59pm

Your goal for this homework is to induce a sentiment phrasal lexicon and use it to perform Sentiment Analysis, following the approach as described in the paper:

Turney, Peter D. 2002. "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews." Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL).

with the change as specified below:

Information Retrieval: instead of using an external search engine to collect hit counts, you should write your own search code to search in the training set, including implementing the "NEAR" operator.

Note that POS tags are needed to generate sentiment prhases. The tagged imbd corpus is provided for this assignment. Each word has two tags, e.g., "thing_NN_I-NP", the first tag is the POS tag and the second tag is for chunking, and you can ignore the second tag for this assignment.

This project is a closely related to PA #2, we will use the same data set and the same 10-fold cross-validation setting. But instead of developing statistical classifiers, we will develop a close-to-deterministic approach but exploit the idea of building a fancy phrasal sentiment lexicon and using statistical measures.

10-fold cross-validation: For each fold, you generate sentiment phrases from the test set and derive their semantic orientation values using the training set. Your final sentiment analysis accuracy is the average of the 10 runs.

It's your choice to reuse part of the python starter code as provided for PA #2.

Key Steps:

(2 Point!) design regular expressions to generate sentiment phrases as specified in the paper. Please show all your regular expressions and examples of sentiment phrases in your report.

(3 Points!) write code to conduct search including implementing the "NEAR" operator. Please paste the relevant part of code in your report.

(3 Points!) calculate the semantic orientation for each sentiment phrase. Please paste the relevant part of code in your report.

(2 Points!) calculate the polarity score for each test review. Please paste the relevant part of code in your report.

Extra Credit I:

(2 Points!) The sentiment analysis performance using your learned sentiment lexicon might be way lower than the level of performance reported in the paper, that is because the training set we use in this assignment is way smaller than the data they considered in their experiments. Can you try to **replace** either or both of the two given seed words "excellent" and "poor", or/and **add** additional seed words, and see if the updated sentiment lexicon can help to improve the performance of sentiment analysis? You can add at most **two** additional seed words with positive sentiments and at most **two** additional seed words with negative sentiments. To earn extra credit, you need to obtain noticeable performance improvements (.5 percent or more) for sentiment analysis of reviews. Please describe the performance improvements and the set of seed words you use in your report.

Extra Credit II: (3 Points!) (quanitfying gender biases in w2vNEWS embeddings)

Learn the algorithms proposed in the following paper.

Bolukbasi, Tolga and Chang, Kai-Wei and Zou, James Y and Saligrama, Venkatesh and Kalai, Adam T. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. Advances in Neural Information Processing Systems.

Read this arxiv version of the paper (https://arxiv.org/pdf/1607.06520.pdf) and implement the algorithms in Section 5.

(1 Point!) Task 1: write code to identify the gender direction as introduced in section 5.1. The 10 seed word pairs are provided in Figure 5. To identify the gender subspace, you should calculate the 10 gender pair difference vectors using the provided word embeddings in `w2v_gnews_small.txt`, and run PCA to get the g vector.

(2 Point!) Task 2: write code to compute the direct bias of occupation words, using the direct bias calculation method described in section 5.2. The occupation word list is provided in `occupation.txt`. Instead of calculating one value for the set of words that indicates the average direct bias of all the occupation words, calculate a value for each occupation word, and report the two occupation words that have the highest direct bias values, as well as the two occupation words that have the lowest direct bias values.

OUTPUT FORMATTING

The results will look something like the figure in the next page. Even if you program using a language different from Python and do not use the starter code, your program output should follow the same structure.

GRADING CRITERIA

Your program will be graded all based on cross validation results on the provided training set, and mainly based on the correctness of your code (please refer back to the first section for how we will weight each task). We understand that you may produce slightly different results if you have implemented some details in a different way. Therefore, please describe enough details in your written report. In addition, we will test your code on a separate test set to make sure it runs properly. For instance, if we found any experimental result has been hard coded, you won't get any credit for that part.

ELECTRONIC SUBMISSION INSTRUCTIONS

(a.k.a. "What to turn in and how to tun in")

You need to submit 2 things:

1. The source code files for your program. Be sure to include all files that we will need to compile and run your program!
2. A report file that includes the following information:
    - how to compile and run your code
    - results and analysis
    - any known bugs, problems, or limitations of your program

   REMINDER: your program **must** compile and run. We will not grade programs that cannot run.

How to turn in:

1. please include everything in a single .zip package, and submit it via canvas.

```
[INFO] Fold 0 Accuracy: 0.500000

[INFO] Fold 1 Accuracy: 0.500000

[INFO] Fold 2 Accuracy: 0.500000

[INFO] Fold 3 Accuracy: 0.500000

[INFO] Fold 4 Accuracy: 0.500000

[INFO] Fold 5 Accuracy: 0.500000

[INFO] Fold 6 Accuracy: 0.500000

[INFO] Fold 7 Accuracy: 0.500000

[INFO] Fold 8 Accuracy: 0.500000

[INFO] Fold 9 Accuracy: 0.500000

[INFO] Accuracy: 0.500000
```

Figure 1: Sample Output for Sentiment Analyzer