Sentiment Analysis of Yelp Reviews

User's Manual

Md Tahmid Rahman Laskar
Yoon Tae Kim

Department of Computer Science and Engineering

York University

Toronto, ON, M3J1P3,

Canada

Overview:

1. Multinomial Naïve Bayes: Python (Scikit-Learn library was used)
2. SVM: Weka (FilteredClassifier with SGD Text. StringToWordVector filter was used)
3. Random Forest: Weka (FilteredClassifier with Random Forest. StringToWordVector filter was used)
4. LSTM: Python (Keras Library was used)
5. BLSTM: Python (Keras Library was used)
6. GRU: Python (Keras Library was used)

Datasets:

1. We have two types of datasets:
   i. .txt file for Python (Separate file for binary and ternary classification)
   ii. .arff file for Weka (Separate file for binary and ternary classification)
2. Initially, we preprocessed the yelp review dataset which was originally in the .json format to select 100000 reviews randomly where each star rating contains 20000 reviews.
   (The original .json file is not uploaded in the EECS server because the size of this file is 4.7 GB. The .json file can be downloaded from here: https://www.yelp.com/dataset)
3. All the preprocessed .txt file and .arff file are uploaded in the EECS server.

Tutorial Video:

1. We have made a video tutorial on how to use FilteredClassifier with StringToWordVector along with other classifiers in WEKA. The video tutorial can be found here.

Preprocessing:

i. Text reviews:
   a. All the stopped words were removed using NLTK.
   b. All the characters were converted to lowercase.
   c. All the punctuation marks were removed.
   d. All the numeric values were removed.

ii. Rating:
   a. For binary classification:
      i. Ratings having 4 and 5 stars were converted to the value 1 (positive).
      ii. Ratings having 1 and 2 stars were converted to the value 0 (negative).
   b. For ternary classification:
      i. Ratings having 4 and 5 stars were converted to the value 1 (positive).
      ii. Ratings having 1 and 2 stars were converted to the value 0 (negative).
      iii. Ratings having 3 stars were converted to the value 2 (neutral).

Description:

1. **Naïve Bayes:**
    a. Binary Classification:
        i. Source code: NaiveBayesBinary.py
        ii. Dataset:
            1. Training:
                a. train2star.txt
                b. valid2star.txt
            2. Test:
                a. test2star.txt
    b. Ternary Classification:
        i. Source code: NaiveBayesTernary.py
        ii. Dataset:
            1. Training:
                a. train3star.txt
                b. valid3star.txt
            2. Test:
                a. test3star.txt
    c. Sample input and output:
        i. Each line contains a review text followed by its sentiment on the next line. 0 indicates negative sentiment, 1 indicates positive sentiment, 2 indicates neutral sentiment.
2. **Random Forest:**
    a. Binary Classification:
        i. Source code: Implemented in Weka using FilteredClassifier with StringToWordVector filter. A tutorial video is attached regarding how to use FilteredClassifier for Sentiment Analysis in Weka.
        ii. Dataset:
            1. Training:
                a. train2starweka.arff
            2. Test:
                a. test2starweka.arff
    b. Ternary Classification:
        i. Source code: Implemented in Weka using FilteredClassifier with StringToWordVector filter. A tutorial video is attached regarding how to use FilteredClassifier for Sentiment Analysis in Weka.
        ii. Dataset:
            1. Training:
                a. train3starweka.arff
            2. Test:
                a. test3starweka.arff
    c. Sample input and output:
        i. .txt dataset is converted to .arff file format. Review text is in string format. Sentiment values are nominal values (0, 1, 2). 0 indicates negative sentiment, 1 indicates positive sentiment, 2 indicates neutral sentiment (for ternary).

3. **SVM:**
    a. Binary Classification:
        i. Source code: Implemented in Weka using FilteredClassifier with StringToWordVector filter. A tutorial video is attached regarding how to use FilteredClassifier for Sentiment Analysis in Weka.
        ii. Dataset:
            1. Training:
                a. train2starweka.arff
            2. Test:
                a. test2starweka.arff
    b. Ternary Classification:
    SVM was not implemented for ternary classification.
    c. Sample input and output:
        i. .txt dataset is converted to .arff file format. Review text is in string format. Sentiment values are nominal values (0, 1). 0 indicates negative sentiment, 1 indicates positive sentiment.

4. **LSTM:**
    a. Binary Classification:
        i. Source code: LSTMBinary.py
        ii. Dataset:
            1. Training:
                a. train2star.txt
            2. Validation:
                a. valid2star.txt
            3. Test:
                a. test2star.txt
    b. Ternary Classification:
        i. Source code: LSTMTernary.py
        ii. Dataset:
            1. Training:
                a. train3star.txt
            2. Validation:
                a. valid3star.txt
            3. Test:
                a. test3star.txt
    c. Sample input and output:
        i. Each line contains a review text followed by its sentiment on the next line. 0 indicates negative sentiment, 1 indicates positive sentiment, 2 indicates neutral sentiment.

5. **BLSTM:**
    a. Binary Classification:
        i. Source code: BLSTMBinary.py
        ii. Dataset:
            1. Training:
                a. train2star.txt
            2. Validation:
                a. valid2star.txt
            3. Test:
                a. test2star.txt
    b. Ternary Classification:
        i. Source code: BLSTMTernary.py
        ii. Dataset:
            1. Training:
                a. train3star.txt
            2. Validation:
                a. valid3star.txt
            3. Test:
                a. test3star.txt
    c. Sample input and output:
        i. Each line contains a review text followed by its sentiment on the next line. 0 indicates negative sentiment, 1 indicates positive sentiment, 2 indicates neutral sentiment.

6. **GRU:**
    a. Binary Classification:
        i. Source code: GRUBinary.py
        ii. Dataset:
            1. Training:
                a. train2star.txt
            2. Validation:
                a. valid2star.txt
            3. Test:
                a. test2star.txt
    b. Ternary Classification:
        i. Source code: GRUTernary.py
        ii. Dataset:
            1. Training:
                a. train3star.txt
            2. Validation:
                a. valid3star.txt
            3. Test:
                a. test3star.txt
    c. Sample input and output:
        i. Each line contains a review text followed by its sentiment on the next line. 0 indicates negative sentiment, 1 indicates positive sentiment, 2 indicates neutral sentiment.

Instructions:

- All the codes of the neural network based algorithms have used Keras embedding. If you want to use GloVe embedding, then instructions were given as comment in each of the codes.
- We used the PyCharm IDE.
- Python 3 was used to implement the codes.
- Anaconda distribution was used for package management.
- If the user wants to test the code with other input data, then the input data should be given in the similar format of how it's given in the dataset provided by us. Otherwise, the user must modify the code to read the input data.
- A tutorial video can be found here regarding how to use the FilteredClassifier in Weka.
- All the codes (along with datasets) of each algorithms were given in separate folders.
- The results of neural network based approaches may sometimes show slightly better/worse result in different runs. The reason behind this is due to the random weight generation during initialization.

THANK YOU