Sentiment Analysis on the IMDB Movie Review Dataset

Data Science Capstone Project
Predictive Modeling Report

Date:

[03/12/2021]

Team Members: 4

Name: Lawrence Love

Name: Gustavo Ferreira

Name: Frank Zhao

Name: Yan Li

[The purpose of this report is to describe the predictive modeling on the data that you have acquired, pre-processed, and explored in DSCI 591.]

1. Define the Predictive Modeling Problem

A. Input: What are the input data and define the input data clearly?
   a. Each sample of input data is a movie review ranging in length from 4 words to 2450 words (before preprocessing).
   b. There are 50,000 samples to be split for training, validation, and testing in accordance with the algorithm being used – LSTM splits for validation during training.
   c. The input data will be preprocessed via lowercasing, special character removal, stopword removal, tokenizing, and lemmatizing.

B. Data Representation: What is the data representation?
   a. Each sample is represented as a string of English text.

C. Output: What are you trying to predict? Define the output clearly.
   a. The goal of the project is to accurately predict the sentiment of a given movie review. The model will take a movie review as input and output a sentiment for the review as either "positive" or "negative".
      i. "This was not a good movie; the acting was bad" - 'negative'
      ii. "I enjoyed the story, and the ending was happy" - 'positive'

2. Predictive Models

A. What are the methods? Give a general introduction of the methods with references
   a. Long Short-Term Memory (LSTM)
      i. LSTM is a deep learning algorithm that uses a variant of Recurrent Neural Networks (RNN) for learning. When processing a paragraph of text, a traditional RNN may leave out important information from the beginning. With LSTM, that information can be retained.
   b. Logistic Regression
      i. Logistic Regression is a statistical model that predicts the relationship between the dependent variable and the independent variable where the independent variable is binary. It can be seen as the probability of an event occurring based on previous data.
   c. Naïve Bayes
      i. In statistics, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. (source: Naïve Bayes Classifier)
   d. Random Forest
      i. it is an ensemble learning algorithm. The basic premise of the algorithm is that building a small decision-tree with few features is a computationally cheap process. If we can build many small, weak decision trees in parallel, we can then combine the trees to form a single, strong learner by averaging or taking the majority vote. (source: Random Forest)

e. Support Vector Machine
    i. In machine learning, support-vector machines (SVMs, also support-vector networks [1]) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. (source: SVM)

B. Describe the methods with a pseudo code using the definitions in Section 1.
    a. LSTM
        i. First, a call state exists where information is held
        ii. Then, there is a previous hidden state that is concatenated with the input, call it *combine*
        iii. *Combine*, gets fed into a forget layer where non-relevant data is removed
        iv. A candidate layer is created using *combine* to hold possible values to be added to the cell state
        v. *Combine* is also fed into the input layer which decideds what data should be added to the new cell state
        vi. After computing the forget layer, candidate layer, and input layer the cell state is calculated using those vectors and the previous cell state
        vii. The output is then computed
        viii. Pointwise multiplying the output and the new cell state gives us the new hidden state
    b. Logistic Regression
        i. Split the TF or Tf-Idf betweet training and test set
        ii. Instanciate the model
        iii. Train the model using training set and labels
        iv. Perform parameter tuning using GridSearch
        v. Train the model with the best parameters
        vi. Predict the values based on the test set
        vii. Evalueate accuracy, f1, precision and recall
    c. Naïve Bayes
        i. Convert words to a matrix of token counts by "CountVectorizer"
        ii. Call NBClassifier
        iii. Fit_transform training data to model
        iv. Predict test label
        v. Evaluate results (Accuracy, Precision, Recall, F1 scores)
        vi. Confusion matrix
        vii. Tuning: various ngrams, alpha
    d. Random Forest:

Precondition: A training set **S** := $(x_1, y_1), \ldots, (x_n, y_n)$, features **F**, and number of trees in forest **B**.

1 **function** RandomForest(**S** , **F**)

2 **H** ← ∅

3 **for** i ∈ 1, . . . , **B** do

4 $\mathbf{S}^{(i)} \leftarrow$ A bootstrap sample from **S** # we select a bootstrap sample from S where $S^{(i)}$ denotes the *i*th bootstrap.

5 $\mathbf{h_i} \leftarrow$ RandomizedTreeLearn($\mathbf{S}^{(i)}$, **F**) # we then learn a decision-tree using a modified decision-tree learning algorithm

6 $\mathbf{H} \leftarrow \mathbf{H} \cup \{\mathbf{h_i}\}$

7 **end for**

8 **return H**

9 **end function**

10 **function** RandomizedTreeLearn(**S** , **F**)

11 At each node:

12 **f** ← very small subset of **F** # at each node of the tree, instead of examining all possible feature-splits, we randomly select some subset of the features $f \subseteq F$.

13 Split on best feature in **f** # the node then splits on the best feature in f rather than F

14 **return** The learned tree

15 **end function**

e. Support Vector Machine
    i. Convert comment to TF-IDF matrix
    ii. Apply GridSearchCV(SVC(), param_grid, verbose = 3, n_jobs = -1)
    iii. Predict testing labels
    iv. Confusion matrix

C. Justify the choice of the method.
a. LSTM
    i. This method is widely known as one of the best methods available for conducting any kind of language processing due to its ability to retain information from the beginning of a sequence. With this in mind, we knew it was one of the methods we needed to choose.
b. Logistic Regression, Naïve Bayes, Random Forest, Support Vector Machine
    i. These methods fall into more of the traditional sense of "Machine Learning". With LSTM and RNN being Deep Learning methods, we wanted to use these other more traditional methods to compare the performance. Additionally, Deep Learning methods require more powerful machines with extra compute power and GPU usage. These four methods above aren't as demanding and should be considered when resources are limited.
c. CNN

i. For the sake of comparison and satisfying of our curiosity, we wanted to run a basic CNN and compare it to LSTM.

3. Evaluations

A. What metrics do you use for evaluation?
   a. Accuracy
   b. Precision
   c. Recall
   d. F1-Score

B. What is your ground truth?

For our dataset, the labels are 'positive' and 'negative'. And, we have a balanced dataset which means that we have 25,000 instances with 'positive' labels and 25,000 instances with 'negative' labels.

C. Discuss the performance and the limitation of the method.

a. Long Short-Term Memory (LSTM)

Optimizer	RMSprop=.01	RMSprop=.01	RMSprop=.001	Adam=.001	Adam=.001	Adam=.01
Batch_size	64	16	64	128	128	128
					GLoVe	GLoVe
Accuracy	0.888	0.887	0.892	0.888	0.866	0.860
Precision	0.872	0.878	0.905	0.887	0.866	0.846
Recall	0.910	0.900	0.875	0.889	0.867	0.881
F1-Score	0.890	0.889	0.890	0.888	0.866	0.863

Limitations:

- Loss was increasing after first epoch for models without word embeddings
- Regardless of above, these models still performed better than the models with word embedding
- Future work to be conducted in this are to understand this behavior
- Runtime for models without word embeddings was ~90 minutes or more

b. Logistic Regression

	Term Frequency	TF-IDF	TF-IDF Lemma
Accuracy	0.862	0.864	0.852
Precision	0.862	0.864	0.852
Recall	0.862	0.865	0.852

F1-Score	0.863	0.864	0.853

- Mode performed in similar way using the three different inputs
- The use of lemmatization actually slightly decreased the model performance

c. Naïve Bayes

	With stopwords			Without stopwords		
Stat	Multinomial [2,3], 1	Completement [2,3], 1	Bernoulli [2,3]; 0.1	Multinomial [1,3], 1	Completement [1,3], 1	Bernoulli [1,2], 0.1
Accuracy	0.9016	0.9015	0.8918	0.8778	0.8778	0.8621
Precision	0.9139	0.9137	0.9150	0.8824	08824	0.8963
Recall	0.8885	0.8885	0.8656	0.8740	0.8740	0.8214
F1-Score	0.9010	0.9009	0.8897	0.8782	0.8782	0.8572

- The model performs very well with over 90% accuracy.
- The limitation is while lemmatizing, the lemmatization could only convert plural words to singular. The future work would be converting verbs to present tense.

d. Random Forest:

1) Performance

	Data with Stopwords and Sentiment Score	**Data without Stopwords and Sentiment Score**	**Data with Stopwords and TF-IDF**	**Data without Stopwords and TF-IDF**	**Data with Stopwords and TF**	**Data without Stopwords and TF**
Accuracy	0.774	0.769	0.848	0.852	0.857	0.857
Precision	0.778	0.767	0.848	0.849	0.842	0.847
Recall	0.778	0.782	0.84	0.849	0.872	0.863
F1-Score	0.778	0.775	0.844	0.849	0.857	0.855

2) Limitation: when we choose the full data with TF-IDF as features, we will get more than 100,000 features, so we have to reduce the data size in order to get results and avoiding running the models for couple hours.

e. Support Vector Machine

	Unigram	Bigram	Trigram
Accuracy	0.9044	0.9084	0.8812
Precision	0.8999	0.9021	0.8725
Recall	0.9117	0.9187	0.8950
F1-Score	0.9058	0.9099	0.8836

- The bigrams svm performs the best with the highest accuracy in testing.
- The drawback for SVM is training time expensive, with at least 6.5 hours training. As the tuning parameters increase, the training time would increase dramatically.
- The limitation would be hardware that the more CPU cores to compute, the faster the result we get. But for normal users, it is hard to upgrade the CPU with powerful performance.

Appendix

[Addition materials that are not included in the above sections.]

Reference:

1. confusion_matrix/cf_matrix.py at master · DTrimarchi10/confusion_matrix (github.com)
2. https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/
3.

Table of Contributions

The table below identifies contributors to various sections of this document.

	Section	**Writing**	**Editing**
1	**Predictive Modeling Problem Definition**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**
2	**Predictive Models**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**
3	**Evaluations**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**
4	**Appendix**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**	**Frank. Z, Gustavo Ferreira, Lawrence Love, Yan Li**

Grading

The grade is given on the basis of quality, clarity, presentation, completeness, and writing of each section in the report. This is the grade of the group. Individual grades will be assigned at the end of the term when peer reviews are collected.