STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

Movie Review Sentiment Analysis

CS513 Section B
David Fu, Cynthia Loh, Aakash Irengbam, Aniket Patel

December 7, 2021

About CS513

A course at Stevens reviewing the knowledge of knowledge exploration and understanding data.

Abstract

- There has been a rapid, notable increase of movie reviews written by the viewers. This is thanks to sites like IMDB that encourage users to write feedback.
- While reading reviews is a great way for potential viewers to gauge which movie they would enjoy watching or not, the vast amount of reviews make it extremely difficult and time consuming for a person to read through completely.
- In order to overcome this challenge, we developed a way to automate movie review classification and summarize the average sentiment
- This will allow people to quickly be able to gauge the scores of the reviews

Introduction

- There are different technique in NLP on how to convert letters and words into numerical matrices
- By applying four different classifications technique in ML modeling we try to compare the effectiveness of each
- We then compare the accuracy of each of the different methods to find the most optimal solution for the problem

Naive Bayes

Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

K Nearest Neighbor

KNN: A k-nearest-neighbor algorithm is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.

Convolution Neural Net

CNN is a multi-layered version of Artificial Neural Net that uses Convolution Layers and is used to detect complex features in data.

Recurrent Neural Net

LSTM a specialized form of ANN that has a feedback loop that does not process each individual feature without the context of other feature. So context is important when words are processed

Data Preprocessing Techniques

In Natural Language Processing, there are various way to extract features from sentences to convert them into numerical matrices while making sure the data is clean

Cleaning Techniques

- Removing punctuation, special characters
- Removing Stopwords (The, a, and etc)
- Reduce complexity by getting rid of hyphenated words

Feature Extraction

- Bert Transformer
- TF-IDF

Data Analysis

The dataset we are using contains movie reviews gathered from IMDB as well as their binary sentiment classification of either positive or negative. In total there are fifty-thousand movie reviews and the dataset is split evenly for training and testing data. In the entire dataset, there are no more than 30 reviews from the same movie because reviews for the same movie typically have correlated ratings.

Naive Bayes

Accuracy and Precision using Naive Bayes.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

clf = GaussianNB()
clf.fit(bert_sentence_training_features, y_train)
y_pred=clf.predict(bert_sentence_test_features)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85       337
           1       0.85      0.85      0.85       323

    accuracy                           0.85       660
   macro avg       0.85      0.85      0.85       660
weighted avg       0.85      0.85      0.85       660
```

KNN

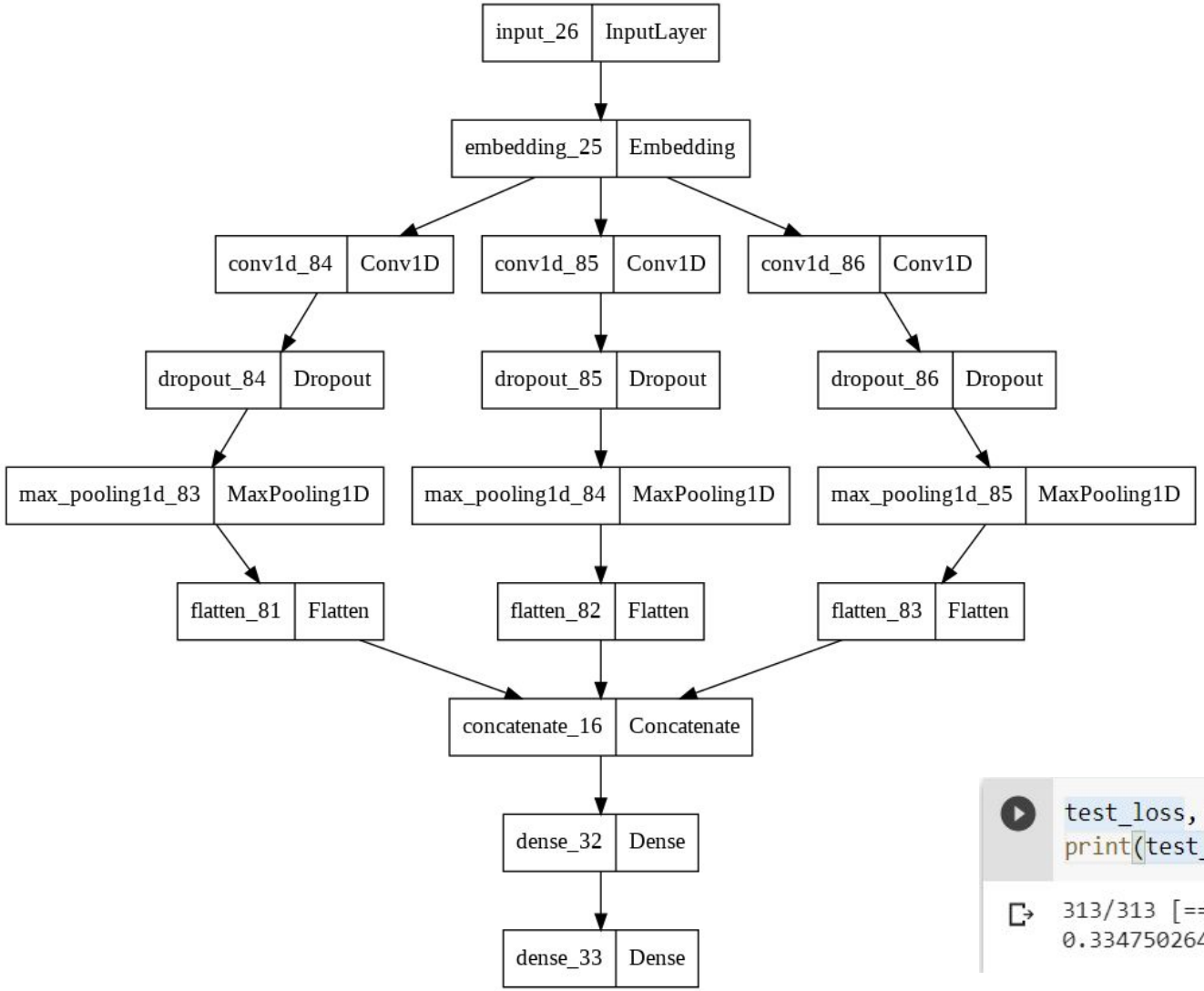
The table below shows the precision, recall, f1-score and support values from using the KNN methodology .

```
[1 0 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0
 1 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0]
Confusion Matrix: [[25  7]
 [ 4 30]]
Accuracy Score:
83.33333333333334
%
              precision    recall  f1-score   support

           0       0.86      0.78      0.82        32
           1       0.81      0.88      0.85        34

    accuracy                           0.83        66
   macro avg       0.84      0.83      0.83        66
weighted avg       0.84      0.83      0.83        66
```

CNN



Accuracy Score: 86.669%

```
test_loss, test_acc = cnn_model.evaluate(X_test, label_test)
print(test_loss, test_acc)
```

```
313/313 [==============================] - 3s 10ms/step - loss: 0.3348 - accuracy: 0.8669
0.3347502648830414 0.8669467568397522
```

RNN - LSTM

Long Short Term Memory Model

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding (Embedding)        (None, None, 50)          4476400

 lstm (LSTM)                  (None, 10)                2440

 dense (Dense)                (None, 2)                 22

=================================================================
Total params: 4,478,862
Trainable params: 4,478,862
Non-trainable params: 0
_________________________________________________________________
```

Embedding Layer LSTM Layer Dense Layer

Activation Function Dropout Layer

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_6 (Embedding)      (None, None, 100)         8952800

 spatial_dropout1d_5 (Spatia  (None, None, 100)         0
 lDropout1D)

 lstm_5 (LSTM)                (None, 100)               80400

 dense_5 (Dense)              (None, 2)                 202

=================================================================
Total params: 9,033,402
Trainable params: 9,033,402
Non-trainable params: 0
_________________________________________________________________
None
```

Model Comparison

	Naive Bayes	KNN	CNN	LSTM
Runtime	Linear	Linear	Linear-Exponential	Exponential
Precision	85%	84%	85.56%	81.81%
Accuracy	85%	83%	86.7%	93.4%
Recall	85%	83%	94.4%	97.6%
Feature	Bert Transformer	Bert Transformer	TFIDF	TFIDF
Complexity	Simple	Medium	Medium-High	Medium-High

Difference in Preprocessing

```
[1 0 1 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0
 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0]
Confusion Matrix: [[25  7]
 [ 4 30]]
Accuracy Score:
83.33333333333334
%
              precision    recall  f1-score   support

           0       0.86      0.78      0.82        32
           1       0.81      0.88      0.85        34

    accuracy                           0.83        66
   macro avg       0.84      0.83      0.83        66
weighted avg       0.84      0.83      0.83        66
```

This is when the preprocessing for KNN is done through the Bert transform.

```
[1. 0. 0. ... 1. 0. 0.]
Confusion Matrix: [[1165 1327]
 [1113 1394]]
Accuracy Score:
51.190238047609526
%
              precision    recall  f1-score   support

         0.0       0.51      0.47      0.49      2492
         1.0       0.51      0.56      0.53      2507

    accuracy                           0.51      4999
   macro avg       0.51      0.51      0.51      4999
weighted avg       0.51      0.51      0.51      4999
```

This is when the KNN analysis preprocessing is done through the simple preprocessing.

Difference in Preprocessing

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

clf = GaussianNB()
clf.fit(bert_sentence_training_features, y_train)
y_pred=clf.predict(bert_sentence_test_features)
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.85	0.85	0.85	337
1	0.85	0.85	0.85	323
accuracy			0.85	660
macro avg	0.85	0.85	0.85	660
weighted avg	0.85	0.85	0.85	660

Preprocessing using TFIDF for naive bayes

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0.0	0.86	0.62	0.73	205
1.0	0.69	0.90	0.78	195
accuracy			0.76	400
macro avg	0.78	0.76	0.75	400
weighted avg	0.78	0.76	0.75	400

Preprocessing using Bert Transformer for naive bayes

stevens.edu

Q&A