MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL

APRIL 2019

PROJECT REPORT

STAR RATING PREDICTION USING MACHINE LEARNING FROM MOVIE REVIEWS

PROJECT MENTOR

Dr. Nilay Khare

SUBMITTED BY

HIMANSHU CHORASIYA	161112037
AMIT KUMAR YADAV	161112068
AJAY KUMAR GAULIA	161112093
RAJENDRA SISODIYA	161112094

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that **Himanshu Chorasiya, Amit Kumar Yadav, Ajay Gaulia and Rajendra Sisodiya** students of B. Tech 3rd Year (Computer Science and Engineering), have successfully completed their project **"Star rating prediction from movie reviews using machine learning "** in partial fulfilment of their Artificial Intelligence project in Computer Science and Engineering.

Dr. Nilay Khare
Assistant Professor
Project Guide

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, hereby, declare that the following report, which is being presented in the Artificial Intelligence Project Documentation entitled **"Star rating prediction using machine learning from movie reviews"** is the partial fulfilment of the requirements of the third year (sixth semester) Artificial intelligence Project in the field of Computer Science and Engineering. It is an authentic documentation of our own original work carried out under the able guidance of Dr. Nilay Khare. The work has been carried out entirely at Maulana Azad National Institute of Technology, Bhopal. The following project and its report, in part or whole, has not been presented or submitted by us for any other purpose in any other institute or organization.

We, hereby, declare that the facts mentioned above are true to the best of our knowledge. In case of any unlikely discrepancies that may possible occur, we will be the ones to take responsibility.

AMIT KUMAR YADAV	**161112068**
HIMANSHU CHORASIYA	**161112037**
RAJENDRA SISODIYA	**161112094**
AJAY GAULIA	**161112093**

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide Dr. Nilay Khare for his valuable help and guidance. We are thankful for the encouragement that he has given us in undertaking this project. His rigorous evaluation and constructive criticism are of great assistance.

We are also grateful to our respected director for permitting us to utilize all the necessary facilities of the college.

Needless to mention is the additional help and support extended by our respected HOD, Dr. Meenu Chawla, in allowing us to use the departmental laboratories and other services.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help.

Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing the much-needed support and encouragement.

CONTENT

ABSTRACT

While the information conveyed through review texts and ratings is easily comprehensible, there is a wealth of hidden information in them that is not immediately obvious. Movie ratings play an important role in tasks such as user movie recommendations, verifying the relationship between user submitted reviews and ratings etc. The ability to predict the rating of a movie would be useful considering these aspects. In this project, we try to focus our task of sentiment analysis on IMDB movie review database. We examine the sentiment expression to classify the polarity of the movie review on a scale of 1(highly disliked) to 10(highly liked) and perform feature extraction and ranking and use these features to train our multi-label classifier to classify the movie review into its correct label.

INTRODUCTION

Movie reviews are an important way to gauge the performance of a movie. While providing a numerical/stars rating to a movie tells us about the success or failure of a movie quantitatively, a collection of movie reviews is what gives us a deeper qualitative insight on different aspects of the movie. A textual movie review tells us about the the strong and weak points of the movie and deeper analysis of a movie review can tell us if the movie in general meets the expectations of the reviewer. Sentiment Analysis is a major subject in machine learning which aims to extract subjective information from the textual reviews. The field of sentiment of analysis is closely tied to natural language processing and text mining. It can be used to determine the attitude of the reviewer with respect to various topics or the overall polarity of review. Using sentiment analysis, we can find the state of mind of the reviewer while providing the review and understand if the person was “happy”, “sad”, “angry” and so on.

LITERATURE SURVEY

The original work on this dataset was done by researchers at Stanford University wherein they used unsupervised learning to cluster the words with close semantics and created word vectors. They ran various classification models on these word vectors to understand the polarity of the reviews. This approach is particularly useful in cases when the data has rich sentiment content and is prone to subjectivity in the semantic affinity of the words and their intended meanings.

METHODOLOGY AND WORK DESCRIPTION

.

3.1 DATA

The dataset contains 14885 examples collected from IMDb where each review is labelled with the rating of the movie on scale of 1-10. These 14885 review and rating are obtained from 100 movies. We applied movie data scrapper to IMDB using which we obtained all the data. This data is divided into three sets. 80% of the data is taken for training the model and 10% for cross validation test which is an important aspect in order to reduce the overfitting and other 10% is used for testing the model that we built.

3.2 MODEL ARCHITECTURE AND TRAINING

The overall task in this project is for classification of reviews as on the range of 1 to 10, where 1 corresponds to highly disliked and 10 corresponds to highly liked and there are other ratings in between. Therefore, for this classification task we explored multiple classification models on above feature representations. We used the models ranging from the simple Logistic Regression to the state of-art SVM Classifier. We also used other classification models like Random Forest Classifier. Apart from these, we also trained the above feature representations on Naïve Bayes' Classifier as this is primarily used in case of text mining in combination with Bag of Words and N-Gram Modelling. We also trained a model based on k-Nearest Neighbours to match the similarity between the reviews and classify them accordingly. For all of the above models, we used sklearn modules by tuning their parameters and not changing their implementations and so we will not go into their theory in this report.

Finally, what we used is called the Recurrent Artificial intelligence model.

3.4 MODULE DESCRIPTION

a) TEXT PREPROCESSING

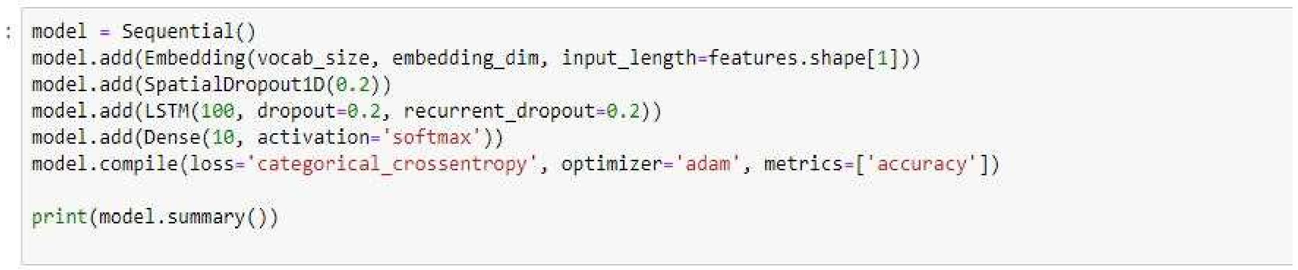
1.Word tokenisation:
Breaking the sentence into individual words called as tokens. We can tokenize them whenever we encounter a space, we can train a model in that way. Even punctuations are considered as individual tokens as they have some meaning.
We have also transformed all the letters into lower case letters.

2.Identification and removal of stop words:
There are various words in the English language that are used very frequently like 'a', 'and', 'the' etc. These words make a lot of noise while doing statistical analysis. We can take these words out. Some NLP pipelines will categorize these words as stop words, they will be filtered out while doing some statistical analysis. Definitely, they are needed to understand the dependency between various tokens to get the exact sense of the sentence.

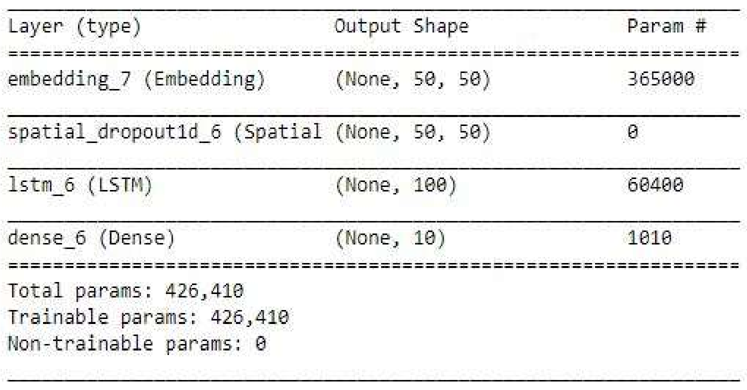
3.Stemming and Lemmatization:
Stemming and Lemmatization are Text Normalization (or sometimes called Word Normalization) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing. Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language. Like for giving, given we can get the root word of both which is 'give' and it provide all the relevant information about these words. Lemmatization is the. Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called **Lemma**. For example the lemma for word "better" will be good.

b) RECURRENT NEURAL NETWORK

Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and belong to the most promising algorithms out there at the moment because they are the only ones with an internal memory.
Because of their internal memory, RNN’s are able to remember important things about the input they received, which enables them to be very precise in predicting what’s coming next.
This is the reason why they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more because they can form a much deeper understanding of a sequence and its context, compared to other algorithms.
Recurrent Neural Networks produce predictive results in sequential data that other algorithms can’t.



```
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, input_length=features.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

print(model.summary())
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_7 (Embedding)      (None, 50, 50)            365000
_________________________________________________________________
spatial_dropout1d_6 (Spatial (None, 50, 50)            0
_________________________________________________________________
lstm_6 (LSTM)                (None, 100)               60400
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1010
=================================================================
Total params: 426,410
Trainable params: 426,410
Non-trainable params: 0
_________________________________________________________________
```

TOOLS AND TECHNOLOGIES USED

5. 1. Software Requirements
   - Python
   - SQL Anaconda Navigator

5. 2. Hardware Requirements
   - Windows XP or Above
   - 2GB of RAM
   - Any Dual Core Processor or above

IMPLEMENTATION AND CODING

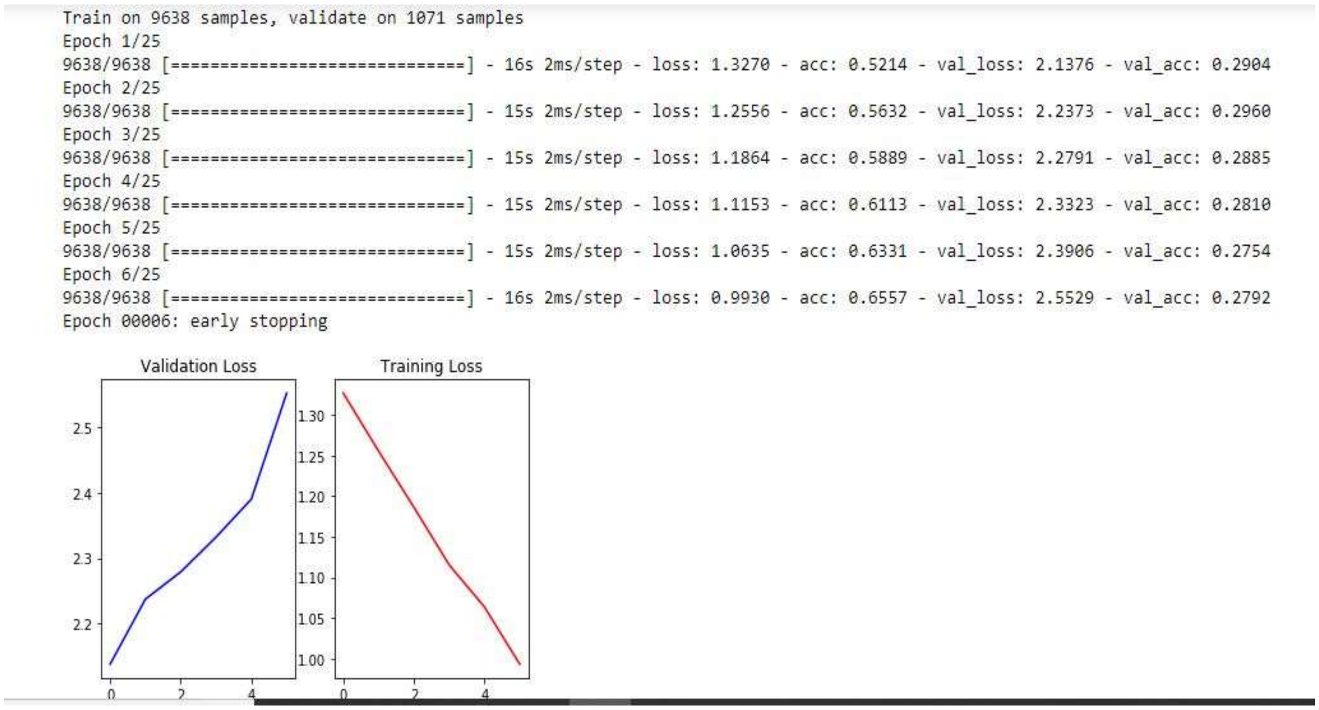
The main challenge of our project was the implementation. After many failed iterations of the idea, we finally landed upon the correct implementation that allowed us to achieve decent performance giving us optimum results.
We used the models ranging from the simple Logistic Regression to the state of-art SVM Classifier. We also used other classification models like Random Forest Classifier.

```
#Importing Libraries and dataset
import numpy as np
import pandas as pd
df = pd.read_csv('datasets/train.csv')
reviews_df = df['review']
ratings_df = df['rating']
reviews = reviews_df.iloc[:].copy()
ratings = ratings_df.iloc[:].copy()
print(reviews.shape, ratings.shape)
print(reviews[0:5])
print(ratings[0:5])

#Text Preprocessing
from string import punctuation
# get rid of punctuation
all_text = ''
for review in reviews:
    # lowercase, standardize
  review = review.lower()
  all_text = all_text + review + '\n'
# removed one extra \n character
all_text = all_text[:-1]
# removed punctuations
all_text = ''.join([c for c in all_text if c not in punctuation])
print(all_text[:50])
# split by new lines and spaces
reviews_split = all_text.split('\n')
all_text = ' '.join(reviews_split)
# create a list of words
words = all_text.split()
words[:30]
# feel free to use this import
from collections import Counter
## Build a dictionary that maps words to integers
counts = Counter(words)
vocab = sorted(counts, key=counts.get, reverse=True)
vocab_to_int = {word: ii for ii, word in enumerate(vocab, 1)}
# for word, ind in vocab_to_int.items():    # for name, age in dictionary.iteritems():  (for
Python 2.x)
#     if ind == 0:
#         print(word)
## use the dict to tokenize each review in reviews_split
## store the tokenized reviews in reviews_ints
reviews_ints = []
for review in reviews_split:
    reviews_ints.append([vocab_to_int[word] for word in review.split()])

# 1=rated one, 0=not reated label conversion
encoded_labels = np.zeros((len(ratings),10))
encoded_labels.shape
for index in range(len(ratings)):
    encoded_labels[index][ratings[index]-1]=1
    encoded_labels = encoded_labels.astype(int)
print('Ratings: ', ratings[:5])
print('\n', encoded_labels[0:5])

# outlier review stats
review_lens = Counter([len(x) for x in reviews_ints])
print("Zero-length reviews: {}".format(review_lens[0]))
print("Maximum review length: {}".format(max(review_lens)))
print('Number of reviews before removing outliers: ', len(reviews_ints))
## remove any reviews/labels with zero length from the reviews_ints list.
```

```
# get indices of any reviews with length 0
non_zero_idx = [ii for ii, review in enumerate(reviews_ints) if len(review) != 0]
# remove 0-length reviews and their labels
reviews_ints = [reviews_ints[ii] for ii in non_zero_idx]
encoded_labels = np.array([encoded_labels[ii] for ii in non_zero_idx])
print('Number of reviews after removing outliers: ', len(reviews_ints), len(encoded_labels))
# pad input so that each review has same length
def pad_features(reviews_ints, seq_length):
    ''' Return features of review_ints, where each review is padded with 0's
        or truncated to the input seq_length.
    '''
# getting the correct rows x cols shape
    features = np.zeros((len(reviews_ints), seq_length), dtype=int)
   # for each review, I grab that review and
    for i, row in enumerate(reviews_ints):
        features[i, -len(row):] = np.array(row)[:seq_length]
    return features

# Test your implementation!
seq_length = 50
# seq_length = max([len(x) for x in reviews_ints])
# print(seq_length)
features = pad_features(reviews_ints, seq_length=seq_length)
## test statements - do not change - ##
assert len(features)==len(reviews_ints), "Your features should have as many rows as reviews."
assert len(features[0])==seq_length, "Each feature row should contain seq_length values."
# print first 10 values of the first 30 batches
print(features[:30,:seq_length])
split_frac = 0.8
## partition data into training, validation, and test data (features and labels, x and y)
split_idx = int(len(features)*split_frac)
train_x, remaining_x = features[:split_idx], features[split_idx:]
train_y, remaining_y = encoded_labels[:split_idx], encoded_labels[split_idx:]
test_idx = int(len(remaining_x)*0.5)
val_x, test_x = remaining_x[:test_idx], remaining_x[test_idx:]
val_y, test_y = remaining_y[:test_idx], remaining_y[test_idx:]
## print out the shapes of your resultant feature data
print("\t\t\tFeature Shapes:")
print("Train set: \t\t{}".format(train_x.shape),
      "\nValidation set: \t{}".format(val_x.shape),
      "\nTest set: \t\t{}".format(test_x.shape))

# Instantiate the model with hyperparams
vocab_size = len(vocab_to_int)+1 # +1 for the 0 padding + our word tokens
output_size = 10
```

```
embedding_dim = 50 # to avoid one hot encoding, as there are very large size vocab list
hidden_dim = 128 # hidden layers dimension
n_layers = 2 # no of hidden layers
batch_size = 64
epochs = 25

#RNN model implementation
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, input_length=features.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

#For Testing
for i in range(10):
   # print(x.shape, y.shape)
    pred = model.predict(test_x[i:i+1])
    # print(pred)
    print()
    print('Actual: ', test_y[i:i+1])
    print('Predicted: ', np.argmax(pred)+1)
```

RESULT ANALYSIS



```
Train on 9638 samples, validate on 1071 samples
Epoch 1/25
9638/9638 [==============================] - 16s 2ms/step - loss: 1.3270 - acc: 0.5214 - val_loss: 2.1376 - val_acc: 0.2904
Epoch 2/25
9638/9638 [==============================] - 15s 2ms/step - loss: 1.2556 - acc: 0.5632 - val_loss: 2.2373 - val_acc: 0.2960
Epoch 3/25
9638/9638 [==============================] - 15s 2ms/step - loss: 1.1864 - acc: 0.5889 - val_loss: 2.2791 - val_acc: 0.2885
Epoch 4/25
9638/9638 [==============================] - 15s 2ms/step - loss: 1.1153 - acc: 0.6113 - val_loss: 2.3323 - val_acc: 0.2810
Epoch 5/25
9638/9638 [==============================] - 15s 2ms/step - loss: 1.0635 - acc: 0.6331 - val_loss: 2.3906 - val_acc: 0.2754
Epoch 6/25
9638/9638 [==============================] - 16s 2ms/step - loss: 0.9930 - acc: 0.6557 - val_loss: 2.5529 - val_acc: 0.2792
Epoch 00006: early stopping
```

REFERENCES

[1] Sentiment Analysis – Wikipedia – https://en.wikipedia.org/wiki/Sentiment_analysis
[2] Large Movie Review Dataset – http://ai.stanford.edu/~amaas/data/sentiment/
[3] Andrew L Mass, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng and Christopher Potts (2011). Learning Word Vectors for Sentiment Analysis
[4] Internet Movie Database – http://www.imdb.com/

[5]CrossValidationWikipediahttps://en.wikipedia.org/wiki/Crossvalidation_%28statistics%29
[6] NLTK Stopwords Corpus: http://www.nltk.org/book/ch02.html
[7] Natural Language Processing from Scratch http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35671.pdf
[8] Scikit-learn API Reference: http://scikit-learn.org/stable/modules/classes.html