IMDB Reviews Rating Prediction and Review Generation

Lavish Gulati **Vakul Gupta** **Chirag Gupta** **Kartik Gupta**
Indian Institute of Technology Guwahati
Group No: 15
{gulat170123030, vakul170101076, gupta170101019, gupta170101030}@iitg.ac.in

Abstract

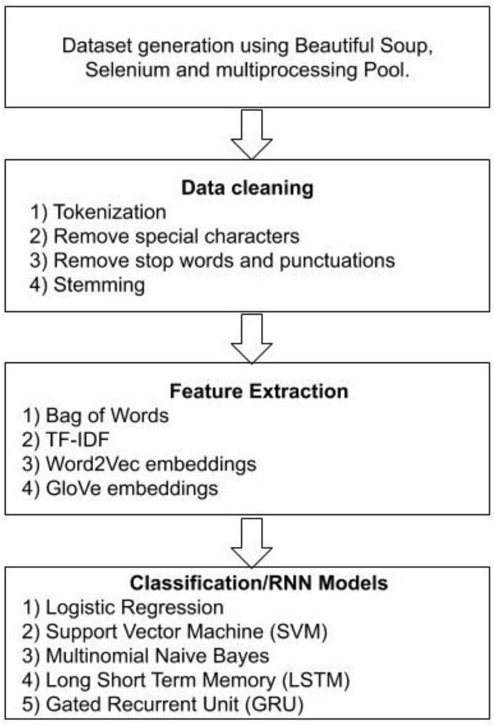
In this study, we explore various natural language processing (NLP) methods to perform sentiment analysis of user reviews on IMDB rated movies, and generate user reviews given movie and rating. We also propose a method to generate an extensive corpus containing the IMDB data and use it to train and analyze various models using feature extraction, classification techniques and neural models to achieve the aforementioned tasks. We approach the task as a multi-class classification with rating outputs as integers from 0 to 10, and use feature extraction methods, including bag of words, tf–idf, Google pretrained word2vec embeddings, and GloVe Twitter pretrained embeddings followed by various classifiers, including multinomial Naive Bayes, SVM, logistic regression, and recurrent neural models, including LSTM and GRU. We have maintained the project code in this GitHub repository.

1 Introduction

Sentiment analysis is a well-known task in the realm of natural language processing. Given a set of texts, the objective is to determine the polarity of that text. (1) provides a comprehensive survey of various methods, benchmarks, and resources of sentiment analysis and opinion mining. Movie reviews are an important way to gauge the performance of a movie. While providing a numerical 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. . 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. In this project, we aim to utilize the relationships of the words in the review to predict the overall polarity of the review. We also aim to generate user reviews given movie and numeric rating.

2 Methodology

The report illustrates a methodology to depict the sentiment analysis of IMDB reviews. First, the generation of a new corpus using web scraping is illustrated. It then feeds the data into the data cleaning module. Then, various feature extraction techniques are applied to transform text into a feature matrix. Last, the report illustrates different algorithms to train and test the feature matrix.


Dataset generation using Beautiful Soup, Selenium and multiprocessing Pool.
Data cleaning
1) Tokenization
2) Remove special characters
3) Remove stop words and punctuations
4) Stemming
Feature Extraction
1) Bag of Words
2) TF-IDF
3) Word2Vec embeddings
4) GloVe embeddings
Classification/RNN Models
1) Logistic Regression
2) Support Vector Machine (SVM)
3) Multinomial Naive Bayes
4) Long Short Term Memory (LSTM)
5) Gated Recurrent Unit (GRU)

2.1 Corpus Generation

A major challenge to IMDB sentiment analysis is an absence of an extensive dataset with a wide array of movies (polarized and neutral) ranging from different timelines, genres and movie industries. To the best of our knowledge, we could only find a dataset (2) generated by Stanford containing a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. Hence we have used the following steps to generate extensive data:

1. **Web scraping using Beautiful Soup:** Beautiful Soup (3) is a Python library for pulling data out of HTML and XML files. It works with lxml parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.
2. **Web browser automation using Selenium:** Selenium (4) is an open-source tool that automates web browsers. A browser-driver then executes these scripts on browser-instance on the system.
3. **Multi-processing using Pool:** The Pool object (5) offers a convenient means of parallelizing the execution of a function across multiple input values, distributing the input data across processes (data parallelism).

2.2 Data cleaning and preprocessing

Data cleaning is the process of detecting and correcting corrupt or inaccurate records from the dataset we have generated. For this we might have to merge few columns or fill up empty records with NaN or pre-defined values.

1. **Denoise text:** Denoising text includes removing special characters and stripping html noise. For the former, a simple regular expression is used ('[^a-zA-z0-9\s]'), and for the latter, BeautifulSoup parser is used to extract the text data which removes any leftover html noise.
2. **Stemming:** nltk (6) Porter Stemmer is based on the idea that the suffixes in the English language are made up of a combination of smaller and simpler suffixes. The group of stems is mapped on to the same stem and the output stem is not necessarily a meaningful word.
3. **Removal of stop words:** Stopwords are the words in any language which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on.

2.3 Feature extraction

Feature extraction involves reducing the number of resources required to describe a large set of data. It is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. We have used the following techniques:

1. **Bag of Words** We have used the CountVectorizer to both tokenize a collection of documents and build a vocabulary, and also to encode new documents using that vocabulary. Using this method, we keep the multiplicity of the words but disregarding grammar and word order.
2. **TF-IDF Vectorisation** The term frequency-inverse document frequency (TF-IDF) is a measure of how a given word is concentrated into relatively few documents (8). Here the terms which appear more frequently are more representative of the content in the documents.
3. **Word2Vec Embeddings** We have used the pre-trained Google word2vec embeddings (9) whose idea is to create a representation of words that capture their meanings, semantic relationships and the different types of contexts they are used in and are capable of boosting the performance of NLP models as they are trained on large News dataset (about 100 billion words).
4. **GloVe Embeddings** The basic idea behind the GloVe word embedding is to derive the relationship between the words from global statistics. We use the pre-trained Stanford word vectors for Twitter (10) trained on 2B tweets and 27B tokens with vocabulary size of 1.2M. Each word representation comprises of a 200-dimensional vector.

2.4 Classification/Neural models

The report implements five classification / neural models to analyze the sentiment of the context given as follows:

1. **Logistic Regression** performs the binary/multi classification by using a sigmoid function as the hypothesis. Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables.
2. **Support Vector Machine (SVM)** The main idea of this algorithm is to map the data from

a relatively low dimensional space to a relatively high dimensional space so that the higher dimensional data can be separated into several classes by a hyper plane.

3. **Multinomial Naive Bayes (MNB)** uses term frequency i.e. the number of times a given term appears in a document. After normalization, term frequency can be used to compute maximum likelihood estimates.

4. **Long Short Term Memory (LSTM)** are capable of learning long-term dependencies. The main advantage of LSTM cell is its cell memory unit which has the ability to encapsulate the notion of forgetting part of its previously stored memory.

5. **Gated Recurrent Unit (GRU)** are a gating mechanism similar to LSTM with a forget gate but lacks an output gate. It's perfomance is similar to LSTM for speech signal modeling and NLP.

3 Progress

3.1 Corpus Generation

This IMDB blog contains a list of 9632 U.S. released movies from 1972 to 2016 divided into 97 pages. Each movie link leads to a separate web page containing user reviews of that movie. Beautiful Soup extracts the list of reviews on a given web page, while Selenium is used to automate opening webpages for each movie and clicking the 'Load More' button to reveal the full list of reviews.
Initially, the Python script tends to run for around 24 hours due to serialized input. To decrease the execution time, we use microprocessing Pool to utilize the quad-core architecture of the CPU, and the execution time is reduced to around 5-6 hours. Finally, we are able to extract 1.7 million user reviews spanning 1.3 GB of system memory.
The generated corpus is accessible here.

3.2 Sentiment Analysis

We divide the different stages of sentiment analysis as different modules in our code structure including different models/algorithms within that module, so that we can directly import the specific module in our Python code. So the project can be used as a tool or library, when working with sentiment analysis problems.

3.2.1 Data cleaing and preprocessing

As a starting step, we read the raw data from the .tsv files and merge the *Comment Body* and *Comment Head* fields to a single field (since keeping both fields separately do not have any significance with respect to sentiment analysis) and finally obtain a Pandas dataframe with two fields: *Text* and *Rating*. Next, we apply data cleaning techniques in the given order:

1. Denoising the text
2. Stemming
3. Removal of stop words

We save the preprocessed data to save redundant computation because this step is common for all the feature extraction and classification techniques.

3.2.2 Feature Extraction

Next step is to read the preprocessed data again into a Pandas dataframe and apply different combinations of techniques to analyze which combination gives better accuracy results. We have applied ten different combinations so far and the corresponding results are displayed in the table.

3.2.3 Architecture of Classification Models

1. **Long Short Term Memory (LSTM)** The LSTM Sequential model comprises of 6 different neural layers. The first layer is formed by a non-trainable *embedding layer* which uses the embedding matrix comprising of all the unique tokens present in the text. The following layer comprises of a *bidirectional LSTM* layer with 32 nodes, which returns the hidden state outputs for all input time steps. Next layer comprises of a *max pooling* layer which downsamples the input representation by taking the maximum value over the time dimension. Next layer comprises of a *dense layer* of 20 neurons with ReLu activation function. We end the sequential model with a *dropout layer* with rate of 0.05, and a *dense layer* of 11 nodes with sigmoid activation function, each node outputting the success probability of each class. The model uses binary crossentropy loss function with adam optimizer.

2. **Gated Recurrent Unit (GRU)** The GRU Sequential model comprises of 3 different neural layers. The first layer is formed by the *embedding layer* same as described in the LSTM model. The following layer comprises of an RNN layer with 100 *GRU nodes*. We end

the sequential model with a *dense layer* of 11 nodes with sigmoid activation function, each node outputting the probability of each class. The model uses binary crossentropy loss function with adam optimizer.

3. **Multinomial Naive Bayes (MNB)** Here we set the additive smoothing parameter as 0.8 and is used for solving the problem of zero probability by a technique used to smooth categorical data.

4. **Logistic Regression (LR)** Here we set the penalty as l2 with the dual formulation variable as true. The tolerance for stopping criterion is 1e-4. The value 0.9 is taken for the inverse of regularization strength. Also, 'newton-cg' is used as solver for optimisation.

5. **Support Vector Machine (SVM)** Again the penalty is chosen as l2 and the tolerance for stopping criterion is 1e-4. The value 0.8 is taken for the inverse of regularization strength. The kernel type used in the algorithm is 'sigmoid'.

3.3 Observations

We are taking top-3 accuracy for comparing the predicted results with the actual results. Top-3 accuracy is defined as follows:
We deduce the predicted rating value given by x to be correct if the actual rating value given by y is such that $x - 1 \leq y \leq x + 1$.
Due to a larger number of classes, top-3 accuracy provides more flexibility since ratings in this window depict a similar sentiment.

Table 1: Classification Models

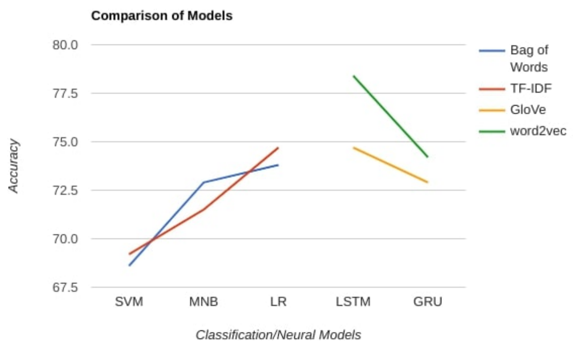
	SVM	MNB	LR
Bag of Words	68.6	72.9	73.8
TF-IDF	69.2	71.5	74.7

Table 2: Recurrent Neural Models

	LSTM	GRU
GloVe	74.7	72.9
word2vec	78.4	74.2

4 Conclusion

We achieve the highest accuracy using the combination of word2vec and LSTM. Relative insensitivity to gap length gives an advantage to LSTM over alternative RNNs and other classification models. Also, SVM does not perform well if the dataset is large and when the target classes are overlapping, both of which are in accordance with our problem. Among feature extraction methods, word2vec and GloVe provide better accuracy than others as they are able to produce more syntactic as well as topically related embeddings. Also, word2vec provides better results than GloVe due to higher dimensionality and larger vocabulary size. The general order of performance for the model is LSTM $>$ GRU $>$ LR $>$ MNB $>$ SVM.



References

[1] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and trends in information retrieval 2, no. 1-2 (2008): 1-135.

[2] Stanford University, *Large Movie Review Dataset*

[3] Chunmei Zheng, Guomei He, Zuojie Peng (2014), A Study of Web Information Extraction Technology Based on Beautiful Soup

[4] Satish Gojarea, Rahul Joshib, Dhanashree Gaigawarec (2015), Analysis and Design of Selenium WebDriver Automation Testing Framework

[5] Navtej Singha, Lisa-Marie Brownea, Ray Butlera, Parallel Astronomical Data Processing with Python: Recipes for multicore machines

[6] Edward Loper, Steven Bird (2002), NLTK: The Natural Language Toolkit

[7] sklearn, *Scikit Learn documentation*

[8] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets (2nd. ed.)*. Cambridge University Press, USA. 8-19, 2014.

[9] Google, *Pretrained word embeddings*

[10] Stanford University, *Pretrained GloVe embeddings*