

CS 410 - Text Information Systems

Final Project Report (Fall 2021)

Team: Butter Chicken

GitHub: [Link](#)

Free Topics | Sentiment Analysis and Topic Modelling on Movie Reviews

Description

We are creating the movie reviews classifier, where we will predict the sentiment of a movie based on the text data that we will extract from social media websites, for eg Twitter. Also, we will do some analysis on the sentiments of different people for the same movie. Sentiments analysis plays a very important role for understanding what people, critics etc think about the movie, and it gives a clear picture to the producer, director, cast of the film and other people who are planning to watch a movie a some sense of how the movie is and what people think about it. It plays an important role while measuring the success rate of movies and what are the things that the audience likes and what can be changed. Also, to get a more clear picture we also also do some topic modelling for the sentiments that were positive and also the sentiments that were negative.

Task:

The main goal of this research is to use textual information to determine the underlying attitude of a movie review. In this research, we attempt to classify if a person enjoyed a film based on the review they leave for it. This is especially beneficial when a movie's creator wants to assess the film's overall performance based on reviews from reviewers and moviegoers. Also, after doing the analysis on sentiments, we are planning to do the topic modelling on the reviews to get a better essence of what people think about the movie.

Value:

It will help the cast, directors and people who are planning to watch the movie about the sentiment of that movie on social media which can influence the success rate of a particular movie. This model will currently be classifying the sentiments of movie review, later we can add the capability to classify the sentiment on any product and could be extended for many other use cases. This project's output can also be utilised to develop a recommender, which provides movie recommendations to viewers based on their prior reviews. Another use for this project would be to locate a group of people who like the same movies.

Approach

1. We are using the Stanford's IMDB movie reviews labelled dataset.
2. We have done preprocessing on the reviews to make it compatible with the classifier's input.
3. After data preprocessing we have created two sentiment analysis classifiers for the same, one is CNN and other is RNN using PyTorch.
4. Also, we have done topic modelling on the same dataset to get better understanding of positive and negative reviews.
5. Then we have scraped the movie reviews from IMDB that the model is not trained on for the evaluation part, here we have used 14 reviews of Avengers movie.
6. We have used Cranfield approach, where we have manually given relevance judgement on the scraped 14 reviews of Avenger movie.
7. At last, we have calculated the accuracy and loss of the models on test data:
 - a. One test accuracy is calculated from the test dataset that is taken from 50000 IMDB reviews.
 - b. Other test accuracy is calculated on the reviews that we have scraped from IMDB.

Changes from Proposal

In our proposal, we added that we will create an API for the sentiment analysis classifier, but the scope of the project changed as we wrongly predicted the time for preprocessing and model creating and training, it took more time than expected and we have removed the API feature from our proposal.

Expected Outcome:

We are expecting our model to tell us the sentiment of a movie based on its text reviews and it will be a binary classifier, which will classify the review as positive or negative and it will tell us the most common keywords used for both positive and negative reviews.

Dataset

In our proposal we thought of exploring the dataset. When we dwell more into Cornell, the data was not gathered at one place, therefore required more time for pre-processing and because of that the scope of the project was getting bigger and we decided to drop it. Though based on our exploration, we can use Cornell data as well along with Stanford dataset.

Stanford's - Large movies review dataset: [Link](#)

For this analysis We will be using the IMDB dataset from Stanford. This dataset of 50,000 movie reviews taken from IMDb. The data is split evenly with 25k reviews intended for training and 25k for testing your classifier. Moreover, each set has 12.5k positive and 12.5k negative reviews. IMDb lets users rate movies on a scale from 1 to 10. To label these reviews the curator of the data labeled anything with ≤ 4 stars as negative and anything with ≥ 7 stars as positive. Reviews with 5 or 6 stars were left out.

Steps to get data:

If you haven't yet, go to [IMDb Reviews](#) and click on "Large Movie Review Dataset v1.0". Once that is complete you'll have a file called `acllmbd_v1.tar.gz` in your downloads folder.

Shortcut: If you want to get straight to the data analysis and/or aren't super comfortable with the terminal, I've put a tar file of the final directory that this step creates here: [Merged Movie Data](#). Double clicking this file should be sufficient to unpack it (at least on a Mac), otherwise `gunzip -c movie_data.tar.gz | tar xopf -` in a terminal will do it.

Unpacking and Merging

Follow these steps or run the shell script here: [Preprocessing Script](#)

1. Move the tar file to the directory where you want this data to be stored.
2. Open a terminal window and `cd` to the directory that you put `acllmbd_v1.tar.gz` in.
3. `gunzip -c acllmbd_v1.tar.gz | tar xopf -`
4. `cd acllmbd && mkdir movie_data`
5. `for split in train test; do for sentiment in pos neg; do for file in $split/$sentiment/*; do cat $file >> movie_data/full_${split}.txt; echo >> movie_data/full_${split}.txt; done; done; done;`

Creating classifier

For this project we have created two classifiers, one is RNN and one is CNN in Pytorch. We got a training accuracy of more than 95% on both the classifiers and we got 75+% for both the classifiers on test data. We have total 50000 labelled reviews and we have used, 40000 reviews for training and 10000 reviews for testing. Also, we have tested the same on the scraped IMDB reviews which is a more realistic testing.

Topic Modelling

Topic modelling refers to the task of identifying topics that best describes a set of documents. These topics will only emerge during the topic modelling process (therefore called latent). And one popular topic modelling technique is known as Latent Dirichlet Allocation (LDA) and we have use LDS to show most popular topics for positive reviews and negative reviews.

Scraper

IMDB data is most sought after when it comes to machine learning. People build recommendation systems, classifiers, and many other ML algorithms and tools on top of it. In this post, we will use Beautiful Soup to scrape data from IMDB.

Attributes that we are interested in:

We will mostly focus on the below-mentioned attributes:

1. Movie title
2. Movie URL
3. Review
4. Review rating

Team Information:

| S.No | Name | NetIDs |
|------|--------------------------------|----------|
| 1 | Ankur Aggarwal | ankura2 |
| 2 | Saniya Wanhar | swanhar2 |
| 4 | Jaskirat Singh Pahwa (Captain) | jpahwa2 |

References:

<https://github.com/Shawn1993/cnn-text-classification-pytorch>

<https://github.com/RaffaeleGalliera/pytorch-cnn-text-classification>

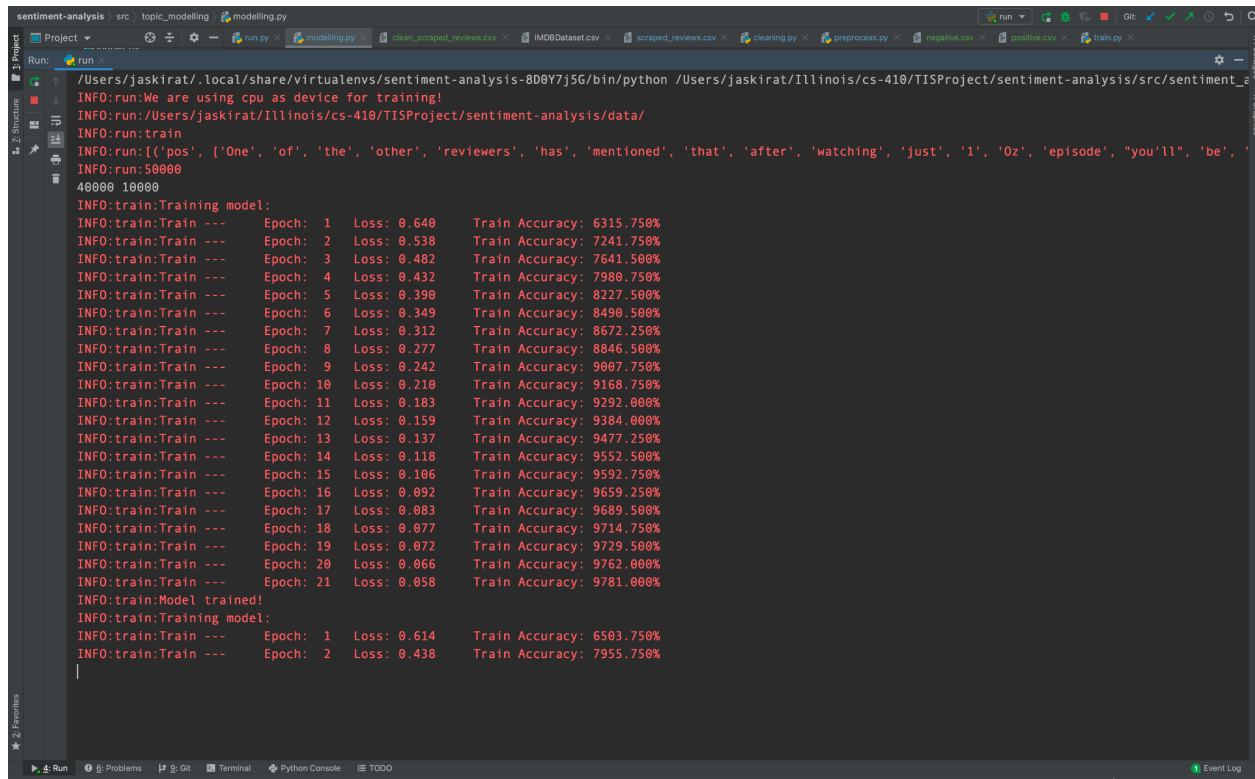
<https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-l-da-35ce4ed6b3e0>

<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/>

<https://dzone.com/articles/predicting-movie-review-sentiment-with-topic-model>

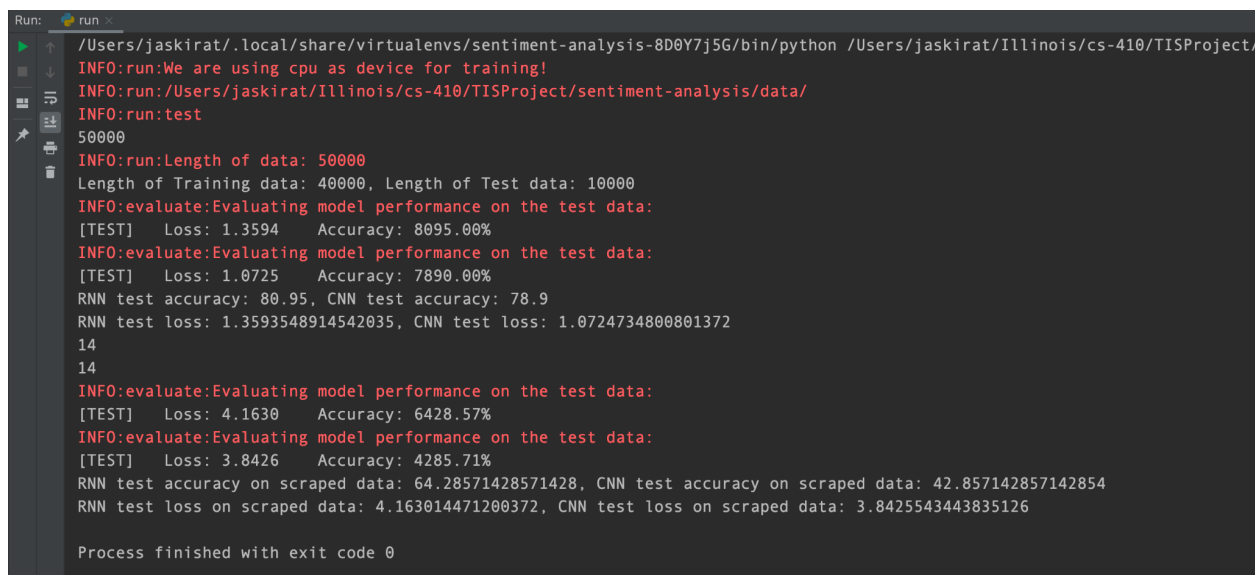
Screenshots:

Training:



```
sentiment-analysis src topic_modelling modelling.py
Project
Run: run
/Users/jaskirat/.local/share/venv/sentiment-analysis-8D0Y7j5G/bin/python /Users/jaskirat/Illinois/cs-410/TISProject/sentiment-analysis/src/sentiment_
INFO:run:We are using cpu as device for training!
INFO:run:/Users/jaskirat/Illinois/cs-410/TISProject/sentiment-analysis/data/
INFO:run:train
INFO:run:[('pos', ['One', 'of', 'the', 'other', 'reviewers', 'has', 'mentioned', 'that', 'after', 'watching', 'just', '1', '0z', 'episode', 'you'll', 'be',
INFO:run:50000
40000 10000
INFO:train:Training model:
INFO:train:Train --- Epoch: 1 Loss: 0.640 Train Accuracy: 6315.750%
INFO:train:Train --- Epoch: 2 Loss: 0.538 Train Accuracy: 7241.750%
INFO:train:Train --- Epoch: 3 Loss: 0.482 Train Accuracy: 7641.500%
INFO:train:Train --- Epoch: 4 Loss: 0.432 Train Accuracy: 7980.750%
INFO:train:Train --- Epoch: 5 Loss: 0.390 Train Accuracy: 8227.500%
INFO:train:Train --- Epoch: 6 Loss: 0.349 Train Accuracy: 8490.500%
INFO:train:Train --- Epoch: 7 Loss: 0.312 Train Accuracy: 8672.250%
INFO:train:Train --- Epoch: 8 Loss: 0.277 Train Accuracy: 8846.500%
INFO:train:Train --- Epoch: 9 Loss: 0.242 Train Accuracy: 9087.750%
INFO:train:Train --- Epoch: 10 Loss: 0.210 Train Accuracy: 9168.750%
INFO:train:Train --- Epoch: 11 Loss: 0.183 Train Accuracy: 9292.000%
INFO:train:Train --- Epoch: 12 Loss: 0.159 Train Accuracy: 9384.000%
INFO:train:Train --- Epoch: 13 Loss: 0.137 Train Accuracy: 9477.250%
INFO:train:Train --- Epoch: 14 Loss: 0.118 Train Accuracy: 9552.500%
INFO:train:Train --- Epoch: 15 Loss: 0.106 Train Accuracy: 9592.750%
INFO:train:Train --- Epoch: 16 Loss: 0.092 Train Accuracy: 9659.250%
INFO:train:Train --- Epoch: 17 Loss: 0.083 Train Accuracy: 9689.500%
INFO:train:Train --- Epoch: 18 Loss: 0.077 Train Accuracy: 9714.750%
INFO:train:Train --- Epoch: 19 Loss: 0.072 Train Accuracy: 9729.500%
INFO:train:Train --- Epoch: 20 Loss: 0.066 Train Accuracy: 9762.000%
INFO:train:Train --- Epoch: 21 Loss: 0.058 Train Accuracy: 9781.000%
INFO:train:Model trained!
INFO:train:Training model:
INFO:train:Train --- Epoch: 1 Loss: 0.614 Train Accuracy: 6593.750%
INFO:train:Train --- Epoch: 2 Loss: 0.438 Train Accuracy: 7955.750%
```

Evaluation:



```
Run: run
/Users/jaskirat/.local/share/venv/sentiment-analysis-8D0Y7j5G/bin/python /Users/jaskirat/Illinois/cs-410/TISProject/
INFO:run:We are using cpu as device for training!
INFO:run:/Users/jaskirat/Illinois/cs-410/TISProject/sentiment-analysis/data/
INFO:run:test
50000
INFO:run:Length of data: 50000
Length of Training data: 40000, Length of Test data: 10000
INFO:evaluate:Evaluating model performance on the test data:
[TEST] Loss: 1.3594 Accuracy: 8095.00%
INFO:evaluate:Evaluating model performance on the test data:
[TEST] Loss: 1.0725 Accuracy: 7890.00%
RNN test accuracy: 80.95, CNN test accuracy: 78.9
RNN test loss: 1.3593548914542035, CNN test loss: 1.0724734800801372
14
14
INFO:evaluate:Evaluating model performance on the test data:
[TEST] Loss: 4.1630 Accuracy: 6428.57%
INFO:evaluate:Evaluating model performance on the test data:
[TEST] Loss: 3.8426 Accuracy: 4285.71%
RNN test accuracy on scraped data: 64.28571428571428, CNN test accuracy on scraped data: 42.857142857142854
RNN test loss on scraped data: 4.163014471200372, CNN test loss on scraped data: 3.8425543443835126

Process finished with exit code 0
```

Topics from Topic Modelling:

Top 20 Positive reviews topic

```
run
[(0,
  '0.006*"see" + 0.005*"time" + 0.005*"well" + 0.005*"really" + 0.005*"even" + '
  '0.004*"like" + 0.004*"good" + 0.004*"get" + 0.004*"first" + 0.003*"would"'),
(1,
  '0.005*"great" + 0.005*"story" + 0.004*"like" + 0.004*"well" + '
  '0.004*"people" + 0.004*"good" + 0.004*"life" + 0.003*"would" + 0.003*"best" '
  '+ 0.003*"time"'),
(2,
  '0.007*"like" + 0.006*"story" + 0.005*"great" + 0.005*"good" + 0.005*"also" '
  '+ 0.004*"love" + 0.004*"even" + 0.004*"time" + 0.004*"well" + '
  '0.004*"would"'),
(3,
  '0.006*"great" + 0.005*"like" + 0.005*"well" + 0.005*"see" + 0.004*"time" + '
  '0.004*"would" + 0.004*"good" + 0.003*"much" + 0.003*"even" + 0.003*"know"'),
(4,
  '0.007*"like" + 0.006*"time" + 0.005*"see" + 0.005*"story" + 0.005*"good" + '
  '0.004*"people" + 0.004*"also" + 0.004*"movies" + 0.004*"think" + '
  '0.004*"much"'),
(5,
  '0.009*"good" + 0.008*"like" + 0.005*"really" + 0.005*"would" + 0.005*"well" '
  '+ 0.005*"see" + 0.005*"story" + 0.005*"time" + 0.004*"great" + '
  '0.004*"first"'),
(6,
  '0.008*"like" + 0.005*"story" + 0.005*"really" + 0.004*"would" + '
  '0.004*"good" + 0.004*"see" + 0.004*"made" + 0.003*"first" + 0.003*"life" + '
  '0.003*"character"'),
(7,
  '0.006*"like" + 0.006*"time" + 0.005*"good" + 0.005*"great" + 0.005*"well" + '
  '0.004*"also" + 0.004*"first" + 0.004*"story" + 0.004*"even" + 0.004*"best"'),
(8,
  '0.006*"like" + 0.005*"story" + 0.005*"well" + 0.004*"see" + 0.004*"great" + '
  '0.004*"love" + 0.003*"also" + 0.003*"really" + 0.003*"time" + 0.003*"way"'),
(9,
  '0.006*"good" + 0.006*"also" + 0.005*"see" + 0.004*"like" + 0.004*"story" + '
  '0.004*"love" + 0.004*"great" + 0.004*"well" + 0.003*"really" + '
  '0.003*"people"')])
```

Top 20 Negative reviews topic

```
DEBUG:gensim.models.ldamodel:bound: at document #0
[(0,
  '0.007*"good" + 0.006*"like" + 0.005*"bad" + 0.005*"even" + 0.005*"well" + '
  '0.005*"time" + 0.004*"really" + 0.003*"character" + 0.003*"acting" + '
  '0.003*"could"'),
(1,
  '0.008*"like" + 0.008*"bad" + 0.005*"even" + 0.005*"movies" + 0.005*"would" '
  '+ 0.005*"get" + 0.005*"time" + 0.004*"good" + 0.004*"see" + 0.004*"really"'),
(2,
  '0.006*"like" + 0.006*"would" + 0.005*"bad" + 0.005*"even" + 0.004*"made" + '
  '0.004*"really" + 0.004*"make" + 0.004*"could" + 0.004*"time" + '
  '0.004*"good"'),
(3,
  '0.008*"like" + 0.005*"really" + 0.005*"would" + 0.005*"even" + 0.005*"get" '
  '+ 0.005*"good" + 0.004*"time" + 0.004*"see" + 0.004*"people" + 0.004*"bad"'),
(4,
  '0.009*"bad" + 0.007*"like" + 0.007*"even" + 0.005*"would" + 0.004*"see" + '
  '0.004*"really" + 0.004*"good" + 0.004*"plot" + 0.004*"made" + 0.004*"time"'),
(5,
  '0.009*"like" + 0.006*"good" + 0.006*"would" + 0.005*"even" + 0.005*"bad" + '
  '0.005*"people" + 0.004*"story" + 0.004*"time" + 0.004*"first" + '
  '0.004*"much"'),
(6,
  '0.008*"good" + 0.007*"like" + 0.006*"really" + 0.005*"much" + 0.005*"even" '
  '+ 0.005*"make" + 0.005*"time" + 0.004*"could" + 0.004*"bad" + 0.004*"see"'),
(7,
  '0.011*"like" + 0.005*"even" + 0.005*"time" + 0.005*"would" + 0.005*"get" + '
  '0.004*"much" + 0.004*"good" + 0.004*"made" + 0.004*"really" + '
  '0.003*"first"'),
(8,
  '0.009*"like" + 0.006*"even" + 0.006*"would" + 0.004*"really" + 0.004*"much" '
  '+ 0.004*"good" + 0.004*"make" + 0.004*"time" + 0.004*"bad" + 0.004*"see"'),
(9,
  '0.006*"like" + 0.005*"story" + 0.005*"good" + 0.004*"bad" + 0.004*"see" + '
  '0.004*"even" + 0.004*"time" + 0.004*"get" + 0.004*"also" + 0.004*"really"')])
```

