Project Report

**Text Information Systems (Fall 2021)**

**Team Butter Chicken**

<to-add -github-link>

<to-add-video presentationlink>

**Free Topics | Sentiment Analysis and Topic Modelling on Movie Reviews**

# Tasks Completed:

* We will be using the IMDB dataset from Stanford, we tried to explore some datasets from different websites but we did not get any other labelled datasets. We found something from cornell but the pre-processing for that dataset will take a lot of time because that is just a scraped dataset with a lot of unnecessary data.
* Then we have divided the dataset into train, test and validation datasets.
* There were some cleaning and preprocessing required in the dataset like removing stop words, adding sentence start and end symbols, converting a review into tokens etc.
* So far, we have created two models, one is CNN and the other is RNN, but training and testing is still left.
* Also, for validation part, we have scraped the the IMDB movie reviews from their website and we have scraped reviews for more than 100 movies.

|  |  |  |
| --- | --- | --- |
| S.N. | Task | Estimated Time (in hours) |
| 1 | Explore the datasets | 6 |
| 2 | Combine all the labelled datasets of movie reviews | 2 |
| 3 | Pre-processing on the dataset | 2 |
| 4 | Explore different Machine Learning and Deep Learning algos to create the classifier | 15 |
| 5 | Topic modelling on the whole dataset | 10 |
| 6 | Scrape the real time movie reviews for some movies for the evaluation from IMDB website | 5 |
| 7 | We will manually find the sentiment from the web for the above scraped movies for evaluation | 5 |
| 8 | Compute the error matrices | 4 |
| 9 | Testing and debugging | 6 |
| 10 | Create an API of the model | 5 |

# Data Overview

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

# Step 1: Download and Combine Movie Reviews

If you haven’t yet, go to [IMDb Reviews](http://ai.stanford.edu/~amaas/data/sentiment/) and click on “Large Movie Review Dataset v1.0”. Once that is complete you’ll have a file called aclImdb\_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](https://github.com/aaronkub/machine-learning-examples/blob/master/imdb-sentiment-analysis/movie_data.tar.gz). 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](https://github.com/aaronkub/machine-learning-examples/blob/master/imdb-sentiment-analysis/preprocess_reviews.sh)

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 aclImdb\_v1.tar.gz in.
3. gunzip -c aclImdb\_v1.tar.gz | tar xopf -
4. cd aclImdb && 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;

# Step 2: Creating classifier

# Step 3: Topic Modelling

Topic Models, in a nutshell, are a type of statistical language models used for uncovering hidden structure in a collection of texts. In a practical and more intuitively, you can think of it as a task of:

**Dimensionality Reduction**, where rather than representing a text T in its feature space as {Word\_i: count(Word\_i, T) for Word\_i in Vocabulary}, you can represent it in a topic space as {Topic\_i: Weight(Topic\_i, T) for Topic\_i in Topics}

**Unsupervised Learning**, where it can be compared to clustering, as in the case of clustering, the number of topics, like the number of clusters, is an output parameter. By doing topic modeling, we build clusters of words rather than clusters of texts. A text is thus a mixture of all the topics, each having a specific weight

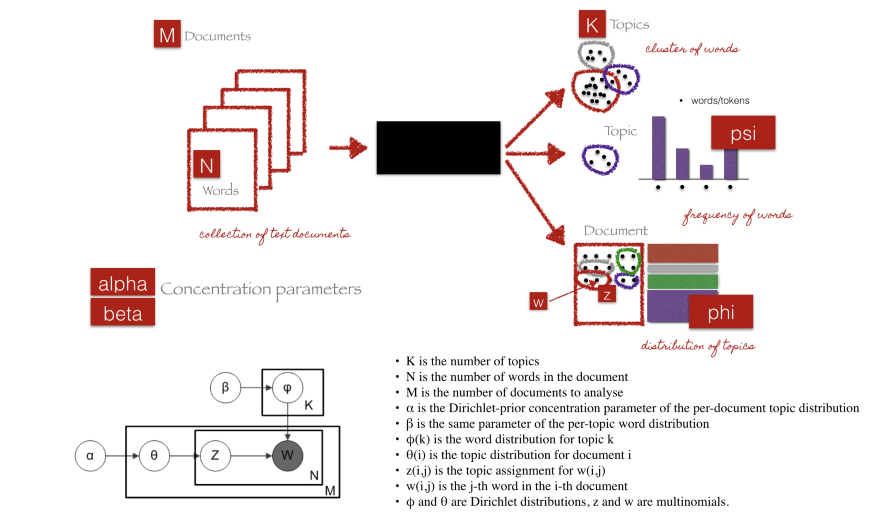
**Tagging**, abstract “topics” that occur in a collection of documents that best represents the information in them.

There are several existing algorithms you can use to perform the topic modeling. The most common of it are, Latent Semantic Analysis (LSA/LSI), Probabilistic Latent Semantic Analysis (pLSA), and Latent Dirichlet Allocation (LDA)

In our project, we have applied LDA, and implemented our topic model using the sklearn implementation in python.

# Theoretical Overview

LDA is a generative probabilistic model that assumes each topic is a mixture over an underlying set of words, and each document is a mixture of over a set of topic probabilities.



Source of image: <http://chdoig.github.io/pytexas2015-topic-modeling/#/3/4>

We can describe the generative process of LDA as, given the M number of documents, N number of words, and prior K number of topics, the model trains to output:

psi, the distribution of words for each topic K

phi, the distribution of topics for each document i

# **Parameters of LDA**

***Alpha parameter****is Dirichlet prior concentration parameter that represents document-topic density — with*a higher alpha, documents are assumed to be made up of more topics and result in more specific topic distribution per document.

***Beta parameter****is the same prior concentration parameter that represents topic-word density —*with high beta, topics are assumed to made of up most of the words and result in a more specific word distribution per topic.

# LDA Implementation

The complete code is mentioned in the Project code. Below are the steps we have followed :

1. Loading data: The output of the sentiment classifier is given as input to the LDA model.
2. Data cleaning: Only the relevant column i.e. “Review text” is taken for further pre-processing and rest columns are dropped. We have used regular expressions to remove any punctuation, and then lowercase the text.
3. Exploratory analysis: We have made a word cloud using the [wordcloud](https://github.com/amueller/word_cloud" \t "_blank) package to get a visual representation of most common words. It is key to understanding the data and ensuring we are on the right track, and if any more preprocessing is necessary before training the model.

This helped us remove certain domain-specific stop words such as “Movie”,”film”,”br” etc in steps ahead.

1. Preparing data for LDA analysis: We started by tokenizing the text and removing stopwords. Next, we convert the tokenized object into a corpus and dictionary.
2. LDA model training: We have build a model with 10 topics where each topic is a combination of keywords, and each keyword contributes a certain weightage to the topic.

# Step 4: Scrapper

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.

First, we need to import Beautiful Soup along with some other packages

# Python Package imports  
import requests  
from bs4 import BeautifulSoup  
from dateutil.parser import parse  
import concurrent.futures  
import pandas as pd

**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

**Scarper**

We already have downloaded the chrome driver and initialized the driver variable, that we will be needing to scrape the webpage.

In the below method, we will be passing in the imdb url of the movie.

This function loops through the h3 element “lister-item-header”, gather the details of the individual reviews of that movie. And then we save them in a csv file

|  |
| --- |
| def scrape\_imdb\_page(dir\_url,driver):      print ('-'\*20,'Scraping directory page','-'\*20)      movie\_links = []      #execute js on webpage to load faculty listings on webpage and get ready to parse the loaded HTML      soup = get\_js\_soup(dir\_url,driver)      for link\_holder in soup.find\_all('h3',class\_='lister-item-header'): #get list of all <div> of class 'name'          rel\_link = link\_holder.find('a')['href'] #get url          x = 'https://www.imdb.com' + rel\_link.replace("?ref\_","reviews/?ref\_")          movie\_links.append(x)      return movie\_links  def scrape\_movie\_page(fac\_url,driver):      print(fac\_url)      soup = get\_js\_soup(fac\_url,driver)      #print(soup)      bio = ''      reviews= []      #profile\_sec = soup.find('div',class\_='lister-item-content')      print('here')        movie\_name = soup.find\_all('div',class\_='parent')[0].find('a').get\_text()      for link\_holder in soup.find\_all('div',class\_='lister-item-content'):          bio = '\"' +  movie\_name + '\"' + ',\"' +  fac\_url + '\"'          try:              bio = bio + ',\"' + process\_bio(link\_holder.find('span',class\_='').get\_text()) + '\"'          except:              print('exception occured')          try:              bio = bio + ',\"' + process\_bio(link\_holder.find('a',class\_='title').get\_text()) + '\"'          except:              print('exception occured')          #print(link\_holder.encode("utf-8"))          try:              bio = bio + ',' + '\"' + process\_bio(link\_holder.find('div',class\_='content').find('div').get\_text()) + '\"'          except:              print('exception occured')          reviews.append(bio)      return reviews |

**Conclusion**

From the above, we can see that it was so easy and straightforward to get the data from IMDB for your analysis. Collection of data along with storage can happen just in a matter of minutes.

**References:**

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

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

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

Team Information:

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