# Sentiment Analysis of Movie Reviews
# Project Report

**CSC 664/864 Multimedia Systems**

**Phyo Htut**
**Sanjay Mirani**
**Jose Pascual**

## Introduction

In this project, we are trying to train a machine learning model that can determine the sentiment of the movie review given a review of the movie. This topic has been a hot research topic for awhile since sentiment analysis in general is quite useful when it comes to social media monitoring. As a result, this will allow us to have a better overview of the public opinion regarding certain topics. This is a challenging topic by nature because it is quite hard even humans to pick up sarcasm from text. Without enough knowledge of the topic, there are many comments that would sound positive but could have been a satirized statement.

## Proposed Problem Formulation

Before collecting data reviews from IMDB, we needed to decide which keywords would represent positive, negative and neutral connotations along with pre and post words that came with it. After analyzing the text's/reviews from IMDB we can cross examine with our annotated text dataset to determine and predict movie ratings based on the text in the review.

## High-Level Description of Our Idea

Our overall goal is to have an app that can determine if a movie is generally good or bad depending on the popular opinion of reviews. Similar to websites like Rotten Tomato and IMDB, which gives out a rating for movies, our app's goal is to simplify everything and get straight to the point of giving users a rating after searching a movie. What makes our app different from the two solidified websites is that our app rates the movies based on text reviews and not the ratings given by the reviewers. There are some reviews that are hypocritical based on the text compared to the rating, and with our app we are able to filter these out because our app is text based. To approach our goal and solve the problem, we started working with a library called NLTK in python, which provides all the tools and methods for natural language processing. NLTK also contained 2000 movie reviews which already separated the positives from the negatives. Working with this library to train our model, we were able to accomplish deciphering texts from our own collected dataset and group them by negative and positive reviews with an accuracy of 85% with our test data and around 90% for our cross validated data. With an 85% accuracy, it means that the model makes a mistake in its prediction once every 7 predictions.

## Description of Our Work

For the backend, we first started with an NLP engine that benchmarked correctness and evaluated the results so we can improve on it. As noted previously, we did go along with the NLTK library with this project. To put it simple, our method design consists of the following steps before the frontend uses it:

- Started off with the NLTK library and collected movie reviews dataset from the reference given in the reference section which comes with pre divided positive and negative movie reviews.
- Applied the data preprocessing techniques like tokenizing, removing stop-words and converting tokens into lowercase.
- Next, dividing this dataset into training and test sets.
- Trained the model with Naive Bayes classifier and saved it.
- Next, we test our classifier on the test dataset just to check over the accuracy.

In addition, we integrated Omdb api to fetch the imdb id for a movie by passing on the movie title. Using api dojo from Rapid api, we were able to get top movie reviews from critics and the users based on the imdb id fetched from the previous api. These reviews are then passed on to the model and sentiment for each review will be displayed. With all of this, the only thing the users have to do is enter a movie name. Our front end is simple and straightforward. We used React.js to build all our front end and communicate to our backend api, Flask (python).

**Section: Back End** (SanjayMirani)
**Section: Front End** (Phyo Htut)
**Section: Data Collection, parts of back end**(Jose Pascual)

## Experimental Evaluations

We started our work with the nltk's built-in movie review dataset but soon realized that just 2000 movie reviews were not enough and started our work on data collection like web-scraping, collecting pre-built dataset, etc. And, finally built up a dataset large and good-enough with a variety of comments. After experimenting on the dataset, we did hit an accuracy of around 90%, but just to make sure we were not overfitting our model, we implemented the K-fold cross validation technique while training the model. And, also added n-gram technique to the model and tested these changes on the testing dataset and hitted an accuracy ratio of around 85%.

After comments ,that are being fetched from the api, passed onto the model, the output shows the positive and negative sentiment of the comment which turns out to be perfect if the comment is strictly biased to one side. But, if it encounters a comment which is a good blend of both the sentiments, it spits out a good balance of both the sentiments and we just pick out the maximum one and display it to the user.

## Conclusion

We trained our model to be as good as we can given the time constraint and our expertise. That said,  we were able to achieve around 80% to 85% accuracy when it comes to sentiment prediction on our cross-validated data. So, it does give out good results but not good enough for it to be making decisions on its own. As previously mentioned, even humans have difficulties while trying to determine the sentiment of comments if they do not have good insight on the topic of the matter. In a way, this model could serve as an aid.

From working on this project, we did learn a lot since we get hands on experience on creating a model. Also, we have a better understanding of how software design plays a huge role when it comes to creating a software. If we start out with a weak design, it becomes almost impossible for us to try and fix the issues later. Because of how we have started training our model, it was not possible for us to try and include TF-IDF while training our model at a later date. In theory, we understand that we have to apply TF-IDF before we create the bag of words but it is just not possible due to how we have pruned our data from the training set. By the time we realized how to successfully integrate TF-IDF to our model, we did not have enough time to restructure our training code.

## Reference

[1] Large Movie Review Dataset. Stanford AI (2011)
https://ai.stanford.edu/~amaas/data/sentiment/