MovieRex

[movierex.net](http://movierex.net/)

Joel Joy

# Generating movie recommendations for a given movie using IMDB plot summaries

Abstract:

The goal of this project is to find an effective and efficient machine learning algorithm to map similarities between movies. The similarity ratings will then be used to provide a list of movies similar to a movie given by the user. The crux of this project lies in finding the most optimal machine learning algorithm through trial and error and comparing results of each machine learning algorithm manually to analyze the quality of the recommendations provided.

Using IMDB plot summaries that are pulled from the web using IMDB adjacent API, a ‘SentenceTransformer’ model can be trained to assign a vector of numerical values to each plot summary. Each plot summary is then assigned a cosine similarity score, comparing every plot summary to every other plot summary. Once the cosine scores have been calculated, the resulting matrix is saved and can then be used to provide a list of ‘Top 10’ recommendations.

Table of Contents:

1. Introduction…………………………………………………………………………..3
2. ML Algorithms………………………………………………………………………..3
3. TFIDF Vectorizer………………………………………………………………...3
4. Transformer………………………………………………………………………4
5. Sentence Transformer………………………………………………………….5
6. Cosine Scoring…………………………………………………………………..5
7. Data Acquisition……………………………………………………………………..6
8. Data Processing……………………………………………………………………..7
9. Tokenization
10. Lemmatization
11. Removing Contractions
12. Removing Proper Nouns
13. Experimentation……………………………………………………………………..8
14. Implementation………………………………………………………………………9
15. Future Work………………………………………………………………………..10
16. Conclusion………………………………………………………………………….11

## Introduction:

The goal of this project is to recommend movies to a user based on plot similarities. Although other recommendation systems exist that use a myriad of factors to determine ‘good’ recommendations, I will only be basing the recommendations on plot summaries and evaluate the results. We will now discuss some of the researched methods and why most of them were ultimately not chosen. As well as the steps taken in this project to achieve the final goal.

Machine Learning Algorithms

The core of this project relies on training an ML model that, when given a movie name, recommends to the user 10 other movies that have similar plots to the given movie. There were several ML models that could achieve the given task and listed below are the ML models that were researched for this purpose.

* TFIDF-Vectorizer

One of the first methods used to measure differences between 2 pieces of texts was a TFIDF-Vectorizer. TFIDF stands for Term Frequency x Inverse Document Frequency and functions exactly as described. Although a tfidfvectorizer is not a machine learning model, it can be very useful in extracting useful information from blocks of text. TF-IDF is a numerical measure used in text mining. The TF-IDF vectorizer converts a collection of documents into a matrix of TF-IDF features.

1. Term Frequency (TF):
2. Inverse Document Frequency (IDF):
3. TF-IDF Weighting:
4. TF-IDF Vectorizer:

* Converts all the documents into a matrix where the rows represent the documents, the columns represent the terms, and the values are the TF-IDF scores.

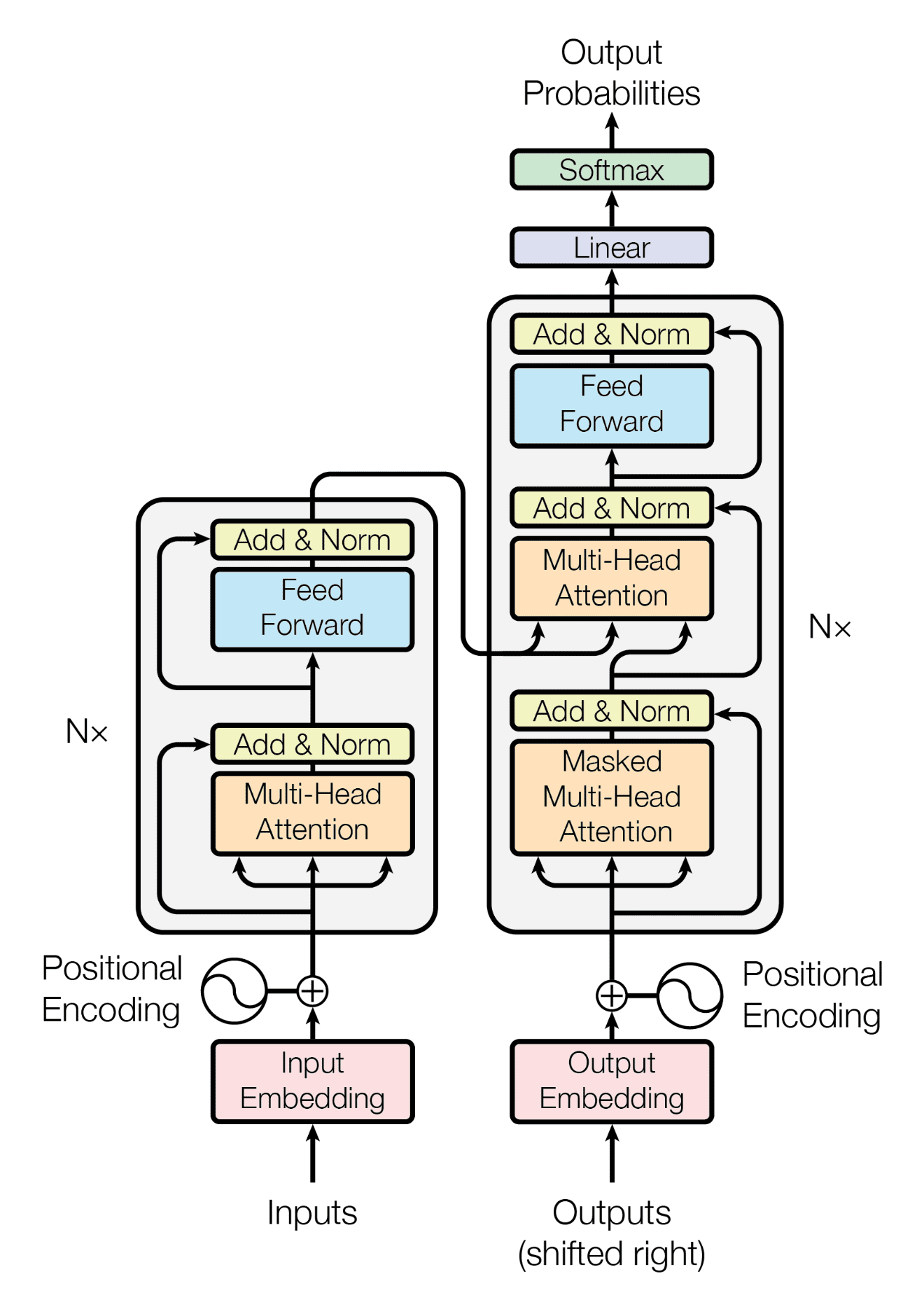
The TF-IDF vectorizer helps to transform a set of text documents into a numerical format that can be used for machine learning tasks, where the importance of each word is represented by its TF-IDF score. This representation is valuable for tasks such as text classification, clustering, and information retrieval.

Although the tfidf-vectorizer is very useful for analyzing documents, it is not very well suited to the task of finding textual similarities between the movie plots. Through some manual testing methods that will be discussed later in the report, it was assessed that the similarity indices assigned by this method were subpar and not acceptable.

* Transformer

After the suboptimal results from using the tfidf-vectorizer, I tried to implement a Transformer model due to its proficiency in Natural Language Processing. Transformer models are a type of deep learning architecture introduced in the paper "Attention is All You Need" by Vaswani et al. in 2017. Transformers have since become the foundation for many state-of-the-art natural language processing (NLP) models and have been successfully applied to various other domains. Here are some of the key features of a Transformer model that make it ideal for processing and analyzing language.

* The core innovation of the Transformer is the self-attention mechanism which allows the model to weigh the importance of different parts of the input sequence when making predictions. Self-attention allows the model to consider all positions in the input sequence simultaneously, capturing relationships between words regardless of their positions.
* Self-attention is calculated using three vectors: Query (Q), Key (K), and Value (V). The attention score is computed as the dot product of Q and K, scaled, and SoftMax applied. The result is then multiplied by the Value vector.
* Since transformers do not inherently understand the order of a sequence, positional encoding is added to the input embeddings to provide information about the position of each element in the sequence.



Sentence Transform:

Sentence transformers refer to a type of transformer-based models that are specifically designed for encoding sentences or short pieces of text into fixed-size vector representations, often referred to as sentence embeddings. The goal is to generate semantically meaningful and context-aware representations of sentences that can be used for various natural language processing tasks.

Similar to other transformer models, sentence transformers are typically pre-trained on large datasets using unsupervised learning objectives. The model is exposed to diverse sentences and learns to encode them into fixed-size vectors.

Cosine score:

Cosine similarity is commonly used to measure the similarity between sentence embeddings. The cosine similarity between two vectors is calculated as the cosine of the angle between them and provides a measure of their similarity.

Data:

One of the challenges faced in this project is the gathering of data required to train the machine learning models. There were 2 realistic ways to acquire the required amount of data:

* Downloading movie data from an existing database:

One of the ways to acquire the large amount of data needed for machine learning in this project would be to access the large and freely accessible database for various datasets on Kaggle. Kaggle is an online resource dedicated to AI and ML communities to share large datasets with each other. Using Kaggle, I was able to download a dataset of movies with approximately 33 thousand entries. The Kaggle movie dataset contained movies from 13 different countries, which then had to be filtered to just the movies from America and Britain, due to the scope of the project. For future expansion of the project, more languages and countries can be considered as sources of movies, but for now, only English movies were considered.

The movie summaries that were acquired from Kaggle were sourced from Wikipedia and were therefore much higher in description length than IMDB plot summaries. After implementing testing methods that will be discussed later in the report, it was determined that having a longer and more descriptive summary of the movies did not actually improve the similarity assessment of the models as compared to the shorter IMDB summaries.

* Using IMDB API to pull movie plots directly from IMDB:

There were multiple ways to gather data from the IMDB website through their API access, even though the official ways to gather data from IMDB are more arduous. After sifting through some online movie databases, I settled upon TMDB (The Movie Database), due to its ease access to its API key when compared to other databases. Once I had access to the api key, I simply used it to download the titles and descriptions of the top 10,000 movies of all time as listed by the database. This database served as a foundation for the data that would be used to train the machine learning models.

In order to further build on this dataset that I had downloaded, I used Wikipedia API to acquire more lengthy and descriptive summaries. Using the list of the top 10,000 movie titles that I had already acquired; I was able to pull the needed movie summaries from Wikipedia. One unseen pitfall of this method, later realized, was that in the case of some movies with names that are simply common English words, the Wikipedia API was not successful in identifying the movie summary.

Data Processing:

After acquiring the database from the various sources available, the data then needed to be processed and pruned so that it could actually be useful in training a machine learning model. There were various text processing techniques that I applied to the plot summaries, some of which were used in the final training process. I tested the processing techniques in conjunction with each other as well as in isolation. Some of the text processing techniques are listed below:

* Tokenization

For the purposes of improving the performance of the machine learning models that I would be training, I decided to apply tokenization to all the plot summaries of the movies that I had accumulated from various sources. Tokenization is the process of breaking down a text into smaller units called tokens. These tokens can be words, subwords, or even characters, depending on the level of granularity required for a particular task.

* Lemmatization

In order to standardize the texts of the plot summaries, I also applied lemmatization on all of the texts. Lemmatization is a natural language processing (NLP) technique that involves reducing words to their base or root form, known as the "lemma." The lemma is the canonical or dictionary form of a word, which represents its core meaning. For example, the words “studies” and “studying” would be converted to the word “study” after undergoing lemmatization. Lemmatization also tends to improve model performance because it treats different inflections of a word as the same feature.

* Removing Contractions

For further standardization of the texts, I used a python function to replace commonly used contractions in the English language with their extended counterparts. Below is a screenshot of my python code that achieves this function:

A screen shot of a computer program

Description automatically generated

* Removing Proper Nouns (Names)

A screen shot of a computer program

Description automatically generated

The above code seeks to filter out proper nouns from the collection of texts and simply replace them with the word “Person”. Using the nltk library, every word in a given sentence can be tagged with its respective part of speech. I then iterate through the list of tags and when encountered by a ‘NNP’ or a ‘NNPS’, the word present at that given index would then be replaced by the term ‘Person’. ‘NNP’ as a tag means that the given word is a proper noun, e.g., America. ‘NNPS’ is also a tag for proper nouns but only when they are plural, e.g., ‘Americans’.

Using a combination of these text processing techniques, I was able to convert the downloaded dataset of movie summaries into a list of texts that were standardized enough to be able to be used by a machine learning model.

Experimentation:

In order to choose between the different machine learning models as well as the different text processing techniques, I had to come up with a way to test the performance of my resulting model. Since the model is supposed to return a list of the top ten movies that have the highest cosine score with the given movie, it would be difficult to measure the validity of the movie recommendations in an ‘objective’ manner. One potential way that I could’ve tested my model would be to use another recommendation algorithm to compare results against. However, I opted against this choice and decided to instead rely on surveys conducted on a small scale.

I created a list of 20 popular movies from different genres, which I then used my model to create recommendations for. Then I created a google form and populated the fields and questions with the list of movies and their respective recommendations.

A screenshot of a computer screen

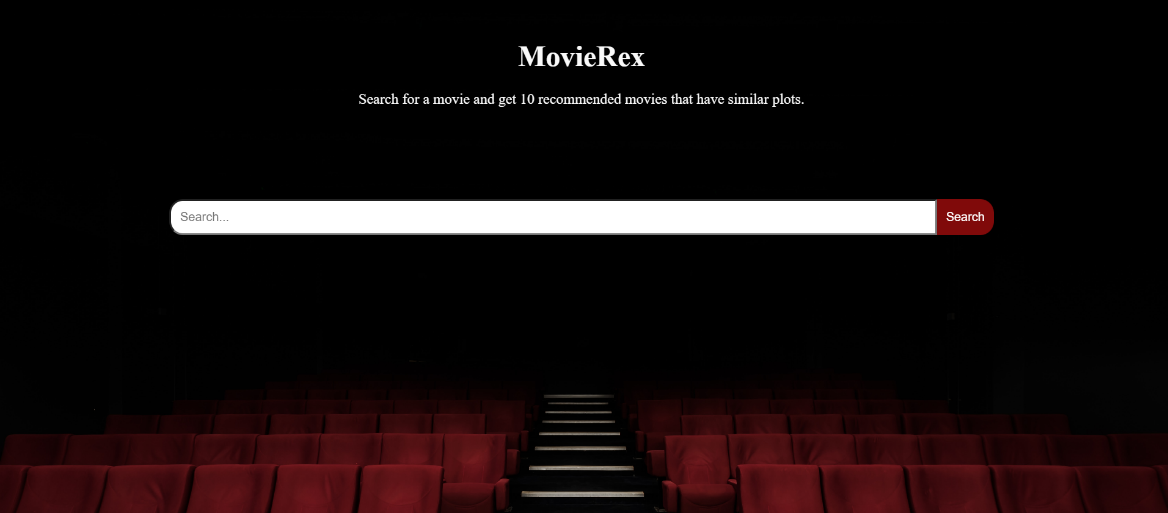
Description automatically generated

The survey was taken by a total of 10 people and their responses were recorded onto an excel spreadsheet. The two groups that the surveys consisted of were: One group where the plot summaries were tokenized, lemmatized, and had contractions removed, and another group where the plot summaries underwent tokenization and name replacement. According to the survey, the former group had an average score of (5.24), while the latter group had an average score of (5.89). Although the difference in scores was not very high, it was significant enough to make decisions about the final implementation.

Implementation:

In the final implementation of the machine learning model, I only utilized some of the text processing techniques as determined by my testing. For the creation of the final model, I used the following text processing techniques: Tokenization, and Name Replacement. The machine learning model that had the best set of recommendations was the Sentence Transformer and was chosen for the final implementation. After the Sentence Transformer vectorized all of the processed plot summaries, the vectors were then compared to each other for similarity using the Cosine Score. The matrix of Cosine Scores were then stored as a file that can be accessed at any time to create a list of recommended movies based on a given movie.

The next step in the implementation was to create a website to host the machine learning model in a presentable manner. I created a rudimentary website and hosted it on DigitalOcean. In order to utilize the Cosine Score Matrix, I created a python application that could be hosted on a web server. The javascript on the website would then send XML requests to the python webserver containing the users input, and the webserver would respond with a list of 10 recommended movies. The UI of the website is shown below:



The UI and design of the website is very simplistic right now as its only purpose is to serve as a convenient medium between the user and the ML algorithm that has been trained to provide movie recommendations. The website can be accessed and tested right now with the given link: <http://movierex.net/>

Future Work:

Many aspects of this project can be further improved upon to create a more polished product. Further experimentation could be conducted with the Sentence Transformer; Formatting and Encoding the data differently could have a huge impact on the vectors created by the transformer, which would then change the similarities between the movies as evaluated by the Cosine Score.

Furthermore, changing the source of the plot summaries could also impact the results of the recommender and lead to potentially better results. For example, if Wikipedia plot summaries are used to train the ML model, it is possible that the vectors would be able to better match the similarities and differences between movies, due to the length of Wikipedia summaries being longer than IMDB summaries. Although more data does not always correspond to better data, the potential of improvement is high for longer, more descriptive plot summaries in my opinion.

The TFIDF vectorizer could be used in conjunction with the currently used Sentence Transformer to create a model that is able to better vectorize the plot summaries of the list of movies. The Sentence Transformer and the TFIDF Vectorizer have been used in conjunction to great effect in the past to find textual similarities in documents.

The website that currently hosts the ML algorithm could definitely be made more user friendly. This could be done by implementing a sort of “Smart Search”, which would try to predict what the user is trying to type into the search bar and provide the user options for auto-completion. This type of “Smart Search” would be very useful to users, especially if they are trying to search for a movie with an especially long name or a name that they do not fully recall.

Conclusion:

This project has been an overall success in terms of the goals that I set out to achieve in the beginning. The ML model that has been trained with the plot summaries of the top 10,000 movies from IMDB provides recommendations for a given movie to a satisfactory level. I plan to improve this project further by implementing some of the changes listed in the ‘Future Work’ section. Working on this project has given me an invaluable learning experience on how to enact a machine learning project and the importance of each step in the process.