Approaches to Industrial Recommendation Systems in Media and Entertainment

Siddharth Mandgi
Applied Artificial Intelligence
Stevens Institute of Technology
NJ, USA
smandgi@stevens.edu

***Abstract*—Recommender Systems simply put, are AI algorithms that utilize features from a reviewed, liked or a purchased product to suggest additional products to consumers. These recommendations can be based on factors such as past purchases, demographic info, their search history, time spent reviewing the product or a like, dislike or a comment left behind by these consumers. The idea of Recommender Systems is that if you can narrow down the pool of selection options for your customers to a few meaningful and relevant choices, they are more likely to make a purchase now, as well as come back for more down the road.This research is a part of the Honors Research Program conducted by Stevens Institute of Technology and I would like to express my sincere gratitude to Prof. Hong Man for being my mentor and guiding me through this project. Through the research topic, we hope to explore about how organizations such as Spotify, Netflix and YouTube leverage Recommender Systems to enjoy a high user share, as well as learn about the essence of approaches such as Natural Language Processing, Neural Networks and LSTM in optimizing user recommendations, making them more relevant and more meaningful.**

***Index Terms*—Recommender Systems, Media, Natural Language Processing, Neural Networks, LSTM**

I. Introduction

A. Motivation

Organizations such as Netflix, YouTube and Spotify have been able to generate a huge viewership with users being hooked to these platforms for hours. Part of this huge viewership is the ability to understand the user's taste on a personal level and recommend content based on this understanding. These tastes vary and can be difficult to understand when it comes to a large audience of millions of users. Recommender Systems are AI algorithms designed to use Machine Learning to understand the kind of content a particular user prefers based on his/her rating, like, dislike or a comment.

The primary motivation for this project lies in understanding how to develop approaches that can generate personalized recommendations that not only meets the users needs but also provides the user an option to explore more.

Recommendation Systems are difficult to implement when it comes to Big Data and though our research we have tried to experiment with different approaches to generate meaningful and relevant recommendations that are powered by smart Machine Learning and Deep Learning models.

Our entire research has been carried out in Python and we have implemented models using Natural Language Processing, Scikit- Learn and LSTM using TensorFlow along with PySpark based ALS model.

B. Project Definition

The goal of this research is to understand how organizations implement recommender systems a large scale to generate meaningful and personalized recommendations for viewers. The objective is to understand features of a particular movie or a music track and recommend movies or tracks which have similar features. We have implemented **Content Based Filtering** and **Collaborative Filtering** methods to achieve highly accurate predictions/recommendations. We have also explored Spotify's recommender systems and how Spotify uses features such as acousticity, valence and tempo, etc. to recommend music based on not only the genre but its beats and emotion. We have also implemented LSTM based neural networks to generate

C. Project Flow

Understanding the dataset: For this Research we have used several datasets including the IMDB dataset, Amazons movies and music dataset and creating our own custom playlist dataset.

Data Preprocessing: Every approach had a different set of prepocessing steps which included cleaning the dataset, creating sentiments and reforming our 'review' feature with spaCy for Content Based Filtering with NLP and counting the no of reviews per movie, popularity threshold, building pivot tables and sparse matrix for Collaborative Filtering.

Training: For training, we have used different algorithms for different approaches. For **Content Based Filtering** we have explored Logistic Regression with spaCy pipelines and LSTM based Neural Nets. For **Collaborative Filtering** we have used K-Nearest Neighbors (KNN) and PySpark based ALS (Alternating Least Squares) model.

Testing: After training our models, we evaluate our predictions on our test datasets. We have also generated music recommendations by training Spotify playlists and movie recommendations on IMDB.

II. Related Work

This section consists of research work from several authors who have worked on recommendation systems using different Machine Learning algorithms. Many of these researches from different authors have aided us in developing our project so far.

A hybrid recommendation technique was proposed in Ghazantar andPragel-Benett [1], and this uses the content-based profile of individual user to find similar users which are used to make predictions. In Sarwar et al. [2], collaborative filtering was combined with an information filtering agent. Here, the authors proposed a framework for integrating the contentbased filtering agents and collaborative filtering. A hybrid recommender algorithm is employed by many applications as a result of new user problem of content-based filtering techniques and average user problem of collaborative filtering [3]. A simple and straightforward method for combining content-based and collaborative filtering was proposed by Cunningham et al. [4]. A music recommendation system which combined tagging information, play counts and social relations was proposed in Konstas et al. [5]. In order to determine the number of neighbors that can be automatically connected on a social platform, Lee and Brusilovsky [6] embedded social information into collaborative filtering algorithm. GroupLens is a news-based architecture which employed collaborative methods in assisting users to locate articles from massive news database[7]. Ringo is an online social information filtering system that uses collaborative filtering to build users profile based on their ratings on music albums[8]. Amazon uses topic diversification algorithms to improve its recommendation[9]. The system uses collaborative filtering method to overcome scalability issue by generating a table of similar items offline through the use of item-to-item matrix. The system then recommends other products which are similar online according to the users' purchase history.On the other hand, content-based techniques match content resources to user characteristics. Content-based filtering techniques normally base their predictions on user's information,and they ignore contributions from other users as with the case of collaborative techniques[10].

These researches have helped us understand the importance of detection of facial key points and challenges associated with it. Upon careful analysis of the different algorithms, we have realized the need to create an efcient model which would correctly predict the facial key points and ensure a highly accurate performance.

III. Implementation

This project is done in various stages illustrated in the project flow. This section will describe these steps in more detail.

A. Dataset Description

We have used Amazon Movies and Music Dataset for Content Based Filtering .

	reviewerID	Movie_ID	reviewerName	helpful	unixReviewTime	reviewTime	reviewText	summary	ratings	sentiment
0	ADZPIG9QOCDG5	0005019281	Alice L. Larson "alice-loves-books"	[0, 0]	1203984000	02 26, 2008	This is a charming version of the classic Dick...	good version of a classic	4	Positive
1	A35947ZP82G7JH	0005019281	Amarah Strack	[0, 0]	1388361600	12 30, 2013	It was good but not as emotionally moving as t...	Good but not as moving	3	Neutral

Fig. 1: Movies Dataset

Our data set consists of a list of 161697533 samples under features such as reviewerID, Movie_ID, reviewerName, ratings, sentiment etc. The most important features to generate recommendations are as follows:-

- reviewerID - User ID used for user to item collaborative filtering and ALS.
- Movie /Music _ID - contains ID's for individual movies or music track.
- ratings - contains ratings on a scale of 5 for every movie/music track.
- sentiment - custom built sentiment column containing three sentiments - Positive, Neutral, Negative for every movie/music track.

We have also used the IMDB reviews dataset containing only reviews.

	review
0	One of the other reviewers has mentioned that ...
1	A wonderful little production. The...
2	I thought this was a wonderful way to spend ti...
3	Basically there's a family where a little boy ...

Fig. 2: IMDB Dataset

We have also used our own custom built playlist using the **spotipy** API

	artist	album	track_name	track_id	danceability	energy	key	loudness	mod
0	Bazzi	COSMIC	Mine	7uzmGiiJyRfuViKKK3lVmR	0.710	0.789	4	-3.874	1
1	ZAYN	Mind Of Mine (Deluxe Edition)	PILLOWTALK	0PDUDa38GO8lMxLCRc4lL1	0.584	0.700	11	-4.275	1

Fig. 3: Spotify Custom Playlist

Features of this dataset:

- Danceability: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength,and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
- Energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.

- Instrumentalness: Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
- Liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live
- Loudness: The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track. Values typical range between -60 and 0 db.
- Speechiness: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.
- Tempo: The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
- Valence: A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful,euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).

B. Approaches to Recommendations

Content Based Filtering with Logistic Regression(spaCy)

A content-based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

In this approach we make use of textual reviews of movies from the Amazon Movie Dataset and apply natural language processing to predict sentiments which is then used to generate recommendations for users.

Data Preprocessing:

- We use spaCy for NLP techniques such as importing stop words, tokenizations and transformation. 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.

```
['further',
 'thereby',
 'we',
 'whenever',
 'thus',
 'becomes',
 're',
 'he',
 'whence',
 'by',
 'our',
 'which',
 'n't',
 'this',
 'too',
```

Fig. 4: Stopwords

- We then build a custom spaCy tokenizer function to create tokens for our reviews and at the same time remove punctuations and convert all words to lower cases.

```
def spacy_tokenizer(sentence):
    mytokens = parser(sentence)
    mytokens = [word.lemma_.lower() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens]
    mytokens = [word for word in mytokens if word not in stopwords and word not in punctuations]
    return mytokens
```

Fig. 5: Tokenizer Function

- Majority of our movies are rated positively with majority of our ratings lying between 4 and 5.

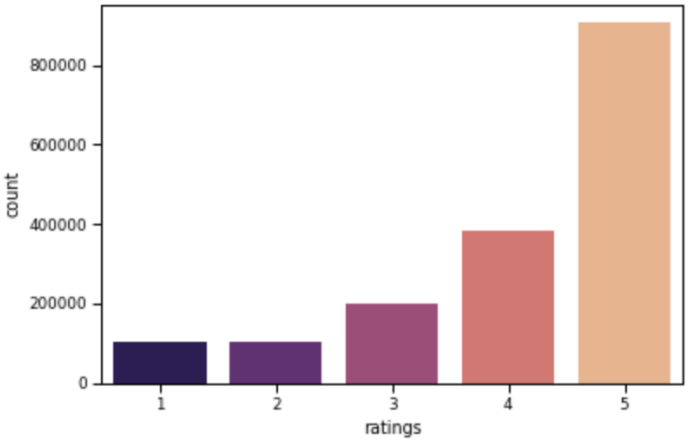

Fig. 6: Seaborn Plot - Ratings

- We then split our data into train and test datasets (where X contains reviews and y contains our ratings/sentiments) and pass it through a Scikit-Learn based Pipeline which contains a tfidf vectorizer , transformer and a Logistic Regression based classifier.

```
pipe = Pipeline([('cleaner', predictors()),
                 ('tfidfVect', tfdifVect),
                ('classifier',classifier),])
```

Fig. 7: Pipeline

- The application of a pipeline is to transform, apply vectorizers and classify every sample of data simultaneously saving time and computational power.

Spotify Features Based Recommendation

In this approach of music recommendation engine, we focus our research on understanding how Spotify analyses different features of music and design a recommender engine for our personal playlists.

Importing our playlists using the spotipy API:

- We require a Client ID and Client Secret which can be obtained by a Spotify Developer Account.
- We then create a custom function to import our playlists from Spotify along with its audio features (such as loudness, valence, acousticity, etc.), artist name, album name and song name.

```
def analyze_playlist(creator, playlist_id):

    # Create empty dataframe
    playlist_features_list = ["artist","album","track_name",  "track_id","danceability","energy","key","loudness","

    playlist_df = pd.DataFrame(columns = playlist_features_list)

    # Loop through every track in the playlist, extract features and append the features to the playlist df

    playlist = sp.user_playlist_tracks(creator, playlist_id)["items"]
    for track in playlist:
        # Create empty dict
        playlist_features = {}
        # Get metadata
        playlist_features["artist"] = track["track"]["album"]["artists"][0]["name"]
        playlist_features["album"] = track["track"]["album"]["name"]
        playlist_features["track_name"] = track["track"]["name"]
        playlist_features["track_id"] = track["track"]["id"]

        # Get audio features
        audio_features = sp.audio_features(playlist_features["track_id"])[0]
        for feature in playlist_features_list[4:]:
            playlist_features[feature] = audio_features[feature]

        # Concat the dfs
        track_df = pd.DataFrame(playlist_features, index = [0])
        playlist_df = pd.concat([playlist_df, track_df], ignore_index = True)

    return playlist_df
```

Fig. 8: Function - Playlist Dataframe

Spotify Features for a playlist:

- We have already spoken about the Spotify features in our dataset description. These features after being rescaled are plotted for a particular playlist.

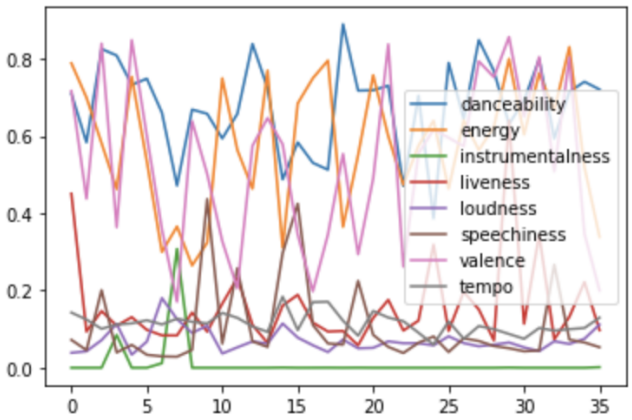

Fig. 9: Spotify Features

Analyzing these features, we derive the following results:

- This playlist suggests that the user likes music that has a high danceability and energy along with songs which have a high tempo.
- Valence suggests if the user has a positive outlook or negative outlook to music.In this case the user likes music that is more positive and livelier.
- Loudness values in this playlist suggests that the user likes softer music.

These results are then used to generate recommendations from Spotify using it's build in recommender engine.

Collaborative Filtering with KNN

Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users.It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. It looks at the items they like and combines them to create a ranked list of suggestions. There are many ways to decide which users are similar and combine their choices to create a list of recommendations. In our approach we have used the Unsupervised Machine Learning Algorithm of clustering with K- Nearest Neighbors.

Data Preprocessing:

- We the create a count variable to count the no of reviews for each movie id along with mean ratings variable to generate the mean ratings for each movie and music track.

Movie_ID	mean_ratings	no_of_reviews
0005019281	4.458716	109
0005119367	4.793478	92
0307141985	4.800000	5
0307142469	4.750000	40
0307142477	2.333333	6
...	...	...

Fig. 10: Creating Variables

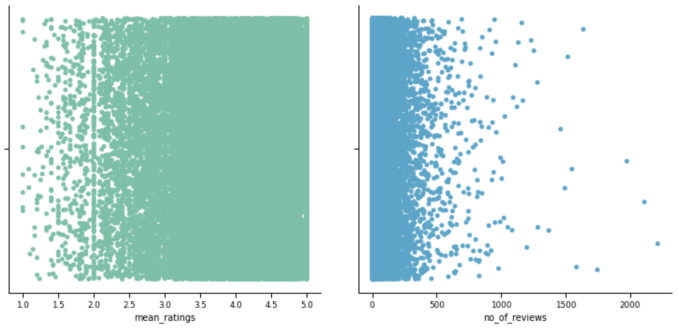

Fig. 11: Mean Ratings and No of Reviews

- From EDA and the above plots, we find that Movie_ID B003EYVXV4 has the max total no of reviews= 2213 and a mean rating = 4.11528 making it the most popular movie.
- We then define a popularity threshold which is a limit below which if the value of the number of ratings fall, they will not be considered.

Pivot Table and Sparse Matrix:

- Pivot tables give each users rating for each movie. The majority of these samples are filled with zeros that will later be replaced by the users predicted rating.

reviewerID	A00295401U6S2UG3RAQSZ	A00348066Q1WEW5BMESN	A0040548
Movie_ID			
0005019281	0.0	0.0	
0005119367	0.0	0.0	
0307142469	0.0	0.0	
0307142485	0.0	0.0	
0307142493	0.0	0.0	
...	...	...	

Fig. 12: Pivot Table

- Pivot table occupies a lot of space when it comes to training a model hence we use sparse matrix which only contains non zero ratings.

K-Nearest Neighbors:

- We use Scikit-Learn's KNN model to train our training dataset and our predictions are computed against a test dataset.

```
from scipy.sparse import csr_matrix as csr
movies_csr = csr(movie_features_df.values)
from sklearn.neighbors import NearestNeighbors as NN
model = NN(metric = 'cosine', algorithm = 'brute')

model.fit(movies_csr)
```

Fig. 13: Sparse Matrix and KNN

IMDB's Weighted Average Algorithm

Often times some movies or music tracks have a very high rating of 4.0 or 5.0 but have very low no of reviews. This implies that the high ratings of track have been provided by very few users and thus its ratings are not completely reliable. To tackle such a situation, we use weighted average ratings which provides ratings considering both the ratings and the no of reviewers

$$W = \frac{Rv + Cm}{v + m}$$

Fig. 14: Weighted Average Algorithm

- W = weighted score
- R = Average Rating
- C = the mean of all votes across the dataset
- v = no of votes or reviews for the movie
- m = minimum votes or number of reviews required to be in the top 250

Content Based Filtering with LSTM

Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They work tremendously well on a large variety of problems, and are now widely used. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way. In our Approach we use LSTM networks for content based content based filtering.

Importing the Data:

- We use make use of the TPU (Tensor Processing Unit) of Google Colab for this approach as it provides us with computational power and speeds up processing
- Hence we need to import our data into a Colab Notebook
- We do this by mounting our Drive to our Colab notebook and reading the csv directly from the directories

Data Preprocessing:

The preprocessing is similar to Content Based Filtering with Logistic Regression approach.

- We create sentiments similarly to our previous approaches.
- We then split the reviews into bag of words while removing punctuations and stop words and count the number of times the words have repeated.

```
[('the', 96908),
 ('and', 48311),
 ('of', 44394),
 ('a', 43601),
 ('to', 37493),
 ('is', 33867),
 ('in', 25817),
```

Fig. 15: Counting Repetitive Words

- Based on these repetitions we form a bag of words by encoding them by their order of repetitions (most to least).

```
version : 101
then : 102
made : 103
think : 104
does : 105
any : 106
over : 107
being : 108
make : 109
```

Fig. 16: Bag of Words

- We then encode the sentiment labels also. 'Positive' labels is encoded as 1 whereas 'Negative' and 'Neutral' is encoded as 0 as we only want the best recommendations possible.
- We then pad our training and testing dataset.
- This is necessary since if one uses the batch mode, all sequences must be of the same length inside your batch.

```
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,   574,     4,    55,     3,
        1431,    36,  1231,    58,   468,    61,   958,     1,    47,
           2,   170,   161,  3149,  3194,  9731,    17, 63169,  4032,
           2, 63170,    42,    63,   197,   359,  1990,    12,     4,
         602,    51,     6,    45,   179,   146,  1392, 12808,    50,
       26630,   100,  3590,    12, 63171,     1,  2476,  2514])
```

Fig. 17: Padded Features

ALS model - PySpark

ALS or the Alternating Least Square Model a matrix factorization algorithm and it runs itself in a parallel fashion. ALS is implemented in Apache Spark ML and built for a larges-scale collaborative filtering problems. ALS is doing a

pretty good job at solving scalability and sparseness of the Ratings data, and it's simple and scales well to very large datasets.

Some high-level ideas behind ALS are:

- Its objective function is slightly different than Funk SVD: ALS uses L2 regularization while Funk uses L1 regularization
- Its training routine is different: ALS minimizes two loss functions alternatively; It first holds user matrix fixed and runs gradient descent with item matrix; then it holds item matrix fixed and runs gradient descent with user matrix
- Its scalability: ALS runs its gradient descent in parallel across multiple partitions of the underlying training data from a cluster of machines

In our approach we have used the ALS model in PySpark to generate Recommendations.

Importing the Data:

- We begin by first creating a spark session and allocating memory size
- We then read the csv into a Spark DataFrame.

```
+--------------+----------+-------+
|    reviewerID|  Movie_ID|ratings|
+--------------+----------+-------+
| ADZPIG9QOCDG5|0005019281|      4|
|A35947ZP82G7JH|0005019281|      3|
|A3UORV8A9D5L2E|0005019281|      3|
|A1VKW06X1O2X7V|0005019281|      5|
|A3R27T4HADWFFJ|0005019281|      4|
|A2L0G56BNOTX6S|0005019281|      5|
```

Fig. 18: Spark DataFrame

Our Model:

- For ALS we need to ensure that all our features are integers. Hence we need to create indexes for reviewerID, Movie_ID and ratings. This is done by a StringIndexer in PySpark. This creates indexes for Movie_ID and reviewerID.

```
+--------------+----------+-------+----------------+--------------+
|    reviewerID|  Movie_ID|ratings|reviewerID_index|Movie_ID_index|
+--------------+----------+-------+----------------+--------------+
| ADZPIG9QOCDG5|0005019281|      4|         88988.0|        3160.0|
|A35947ZP82G7JH|0005019281|      3|        110384.0|        3160.0|
|A3UORV8A9D5L2E|0005019281|      3|        116243.0|        3160.0|
|A1VKW06X1O2X7V|0005019281|      5|         99627.0|        3160.0|
|A3R27T4HADWFFJ|0005019281|      4|         13935.0|        3160.0|
|A2L0G56BNOTX6S|0005019281|      5|         80495.0|        3160.0|
```

Fig. 19: String Indexing

- Hence our new features to fed to the model are - reviewerID_index and Movie_ID _index.
- These features are then used to generate predictions and in turn generate recommendations using RegressionEvaluator and ALS.

IV. RESULTS

Content Based Filtering

The Validation Accuracy for our Logistic Regression model - 0.8526 and we have applied this model to generate recommendations for IMDB reviews.

	review	sentiment	ratings
0	One of the other reviewers has mentioned that ...	Positive	5
1	A wonderful little production. The...	Positive	5
4	Petter Mattei's "Love in the Time of Money" is...	Positive	5
5	Probably my all-time favorite movie, a story o...	Positive	5
6	I sure would like to see a resurrection of a u...	Positive	5
...	...	...	...
208	I just started watching The Show around July. ...	Positive	5
209	This film is well cast, often silly and always...	Positive	5
213	Normally I don't like series at all. They're a...	Positive	5
216	I saw this at the London Film Festival last ni...	Positive	5
218	This movie really woke me up, like it wakes up...	Positive	5

Fig. 20: Top 100 IMDB recommendations

For our model based on LSTM, the is the validation accuracy is 0.769 and takes more computational power and time on local machine.

Spotify Features Based Recommendation

Exploring the features of a playlist we find that user has inclined taste towards tracks that have a high danceability and energy along with tracks which have a high tempo. The user prefers music that is more positive and livelier and is softer. Based on these decisions we have recommended 20 Tracks.

```
Top Recommendations: -
1 - Dixieland Delight - Single Edit - Alabama
2 - I'm the One (feat. Justin Bieber, Quavo, Chance the Rapper & Lil Wayne) - DJ Khaled
3 - Baby Ride Easy (with June Carter Cash) - Johnny Cash
4 - Gentle On My Mind - Remastered 2001 - Glen Campbell
5 - Diggin' Up Bones - Randy Travis
6 - no tears left to cry - Ariana Grande
7 - Sweet Revenge - John Prine
8 - Song of the South - Alabama
9 - Back to You (feat. Bebe Rexha & Digital Farm Animals) - Louis Tomlinson
10 - We'll Meet Again - Johnny Cash
11 - False Alarm - The Weeknd
12 - I Mean It (feat. Remo) - G-Eazy
13 - These Days (feat. Jess Glynne, Macklemore & Dan Caplen) - Rudimental
14 - 7/11 - Beyoncé
15 - Boom Clap - Charli XCX
16 - When You Love Someone - James TW
17 - I Fall To Pieces - Single Version - Patsy Cline
18 - Dark as the Dungeon - Live at Folsom State Prison, Folsom, CA - January 1968 - Johnny
19 - Beautiful (feat. Camila Cabello) - Bazzi
20 - Can't Feel My Face - The Weeknd
```

Fig. 21: Top 20 Spotify recommendations

Collaborative Filtering

KNN gives the most similar (nearest neighbors) items (in our case movies) based on a distance metrics. In our case we have considered a cosine distance metrics. Based on this metric we have generated top 5 recommendations.The best case scenario: Cos0 = 1 which implies maximum similarity and the worst case scenario: Cos90 = 0 which implies maximum dissimilarity. Our distance values are around 0.95 (closer to

```
Recommendations for Movie_ID: 0790744473

1 : 0780623746 with a cosine distance of 0.8951558003262833
2 : 6302993687 with a cosine distance of 0.9205581911370118
3 : 6303824358 with a cosine distance of 0.9345337468831068
4 : 6300251004 with a cosine distance of 0.9531553810066405
5 : 6305513406 with a cosine distance of 0.9585269837602576
```

Fig. 22: Top 5 Recommendations

1) which implies that our recommendations are very similar in features to our reviewed movie.

ALS based PySpark model has lowest computational time and is by the fastest model to train because of parallel computing.The predictions are given below.

reviewerID	Movie_ID	ratings	Movie_ID_index	reviewerID_index	prediction
ATS1HA5M3EPZ4	0783239408	3	148.0	897.0	3.8488202
AKMXCZXE0KD5H	0783239408	5	148.0	3226.0	4.4825144
A1W584W9UW3XEU	0783239408	5	148.0	14423.0	3.5754397
AE0E6UII0VSZA	0783239408	5	148.0	2721.0	4.509235
A2WN6UJP3KUDRH	0783239408	5	148.0	28893.0	4.4495144
AF1VHA7BVWLO1	0783239408	5	148.0	12210.0	4.5768123
AQ4NJ2T9PVI3L	0783239408	5	148.0	2914.0	3.437717
A1Q239XJPSI44W	0783239408	1	148.0	22236.0	2.5796769
AV0OBG2SS1I1R	0783239408	5	148.0	4119.0	4.6192846

Fig. 23: ALS Predictions

Now collaborative recommends items which are most similar and have the closest rating values. Hence our recommendations are given below.

reviewerID_index	recommendations
148	[[22102, 7.525032...]
463	[[42505, 6.882899...]
471	[[49301, 6.460155...]
496	[[49301, 7.932959...]
833	[[48592, 7.792800...]
1088	[[49301, 7.74932]...]
1238	[[47267, 6.549411...]
1342	[[49301, 8.127285...]
1580	[[49301, 7.297943...]
1591	[[47139, 7.110337...]

Fig. 24: ALS Recommendations

IMDB's Weighted Average Algorithm

Using The IMDB weighted average algorithm we generate recommendations based on max weighted average ratings.

	mean_ratings	no_of_reviews	weighted_average
Movie_ID			
B006W9KNXC	4.933835	665	4.903520
B004NSUXHU	4.890323	1085	4.872430
B0007N1BBC	4.924460	278	4.856043
B00006C7G9	4.870321	748	4.845240
B003TO541O	4.876712	511	4.840234
...	...	...	...

Fig. 25: Weighted Average Scores

V. CONCLUSION

Recommender Systems are an important application of AI and through this project I have tried to explore in depth about the different approaches to recommend more personalised music or videos. I have also tried to deep dive methods used by organizations in the Entertainment Industry to recommend content to users to keep them hooked. I hope to continue to explore more about recommender systems in the future.

VI. FUTURE SCOPE

Content Creator's Perspective - So far we have looked at the consumers point of view when it comes to recommendations. A further point of research could be too look at the content creator's perspective for recommendation systems. In the media and entertainment industry which genre of music or a type of music is currently popular among listeners. This can help many young artists reach and appeal to a larger audience. Similarly for movies and videos. AI can help producers gauge the audience interests and help make commercial films which have a larger appeal. **Cloud support** - To generate recommendations for large batches of data having a cloud or server aid could really help carry out speed up processes and generalise our recommendations. **Image Processing** - Images give a lot of information of the data. For example movie posters can be processed and categorized into a particular genre and new movies could be recommended based on these genres.

ACKNOWLEDGMENT

I express my sincere gratitude to Prof Hong Man for his guidance and support under whom I was able to develop this research project. I would also like to thank my advisor Prof Min Song and the ECE Honors Research's Program Director Nina Cheung for giving me this opportunity to showcase my research.

REFERENCES

[1] Burke R. Hybrid web recommender systems. In: Brusilovsky P,Kobsa A, Nejdl W, editors. The adaptive web, LNCS 4321. BerlinHeidelberg, Germany: Springer; 2007. p. 377–408.

[2] Cunningham P, Bergmann R, Schmitt S, Traphoner R, Breen S,Smyth B. "WebSell: Intelligent sales assistants for the World WideWeb. In: Proceedings CBR in ECommerce, Vancouver BC; 2001.p. 104–9.

[3] Konstan I, Stathopoulos V, Jose JM. On social networks andcollaborative recommendation. In: The proceedings of the 32ndinternational ACM conference (SIGIR'09), ACM. New York,NY, USA; 2009. p.195–202.

[4] Lee DH, Brusilovsky P. Social networks and interest similarity:the case of CiteULike. In: Proceedings of the 21st ACMconference on Hypertext and Hypermedia (HT'10). ACM. NewYork, NY, USA; 2010. p. 151–6.

[5] . [30] Condiff MK, Lewis DD, Madigan D, Posse C. Bayesian mixed-effects models for recommender systems. In: Proceedings of ACMSIGIR workshop of recommender systems: algorithm and eval-uation; 1999.

[6] Burke R. Web recommender systems. In: Brusilovsky P, Kobsa A,Nejdl W, editors. The Adaptive Web, LNCS 4321. Berlin Heidelberg (Germany): Springer; 2007. p. 377–408.

[7]] Chen LS, Hsu FH, Chen MC, Hsu YC. Developing recommender systems with the consideration of product profitability for sellers.Int J Inform Sci 2008;178(4):1032–48.

[8] Ziegler CN, McNee SM, Konstan JA, Lausen G. Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web; 2005. p. 22–32..

[9] Min SH, Han I. Detection of the customer time-variant pattern for improving recommender system. Exp System Applications 2010;37(4):2911–22

[10] Schwab I, Kobsa A, Koychev I. Learning user interests throughpositive examples using content analysis and collaborative filtering. Draft from Fraunhofer Institute for Applied Information Technology, Germany; 2001.