Approaches to Recommender Systems in Media & Entertainment

By Siddharth Mandgi.....

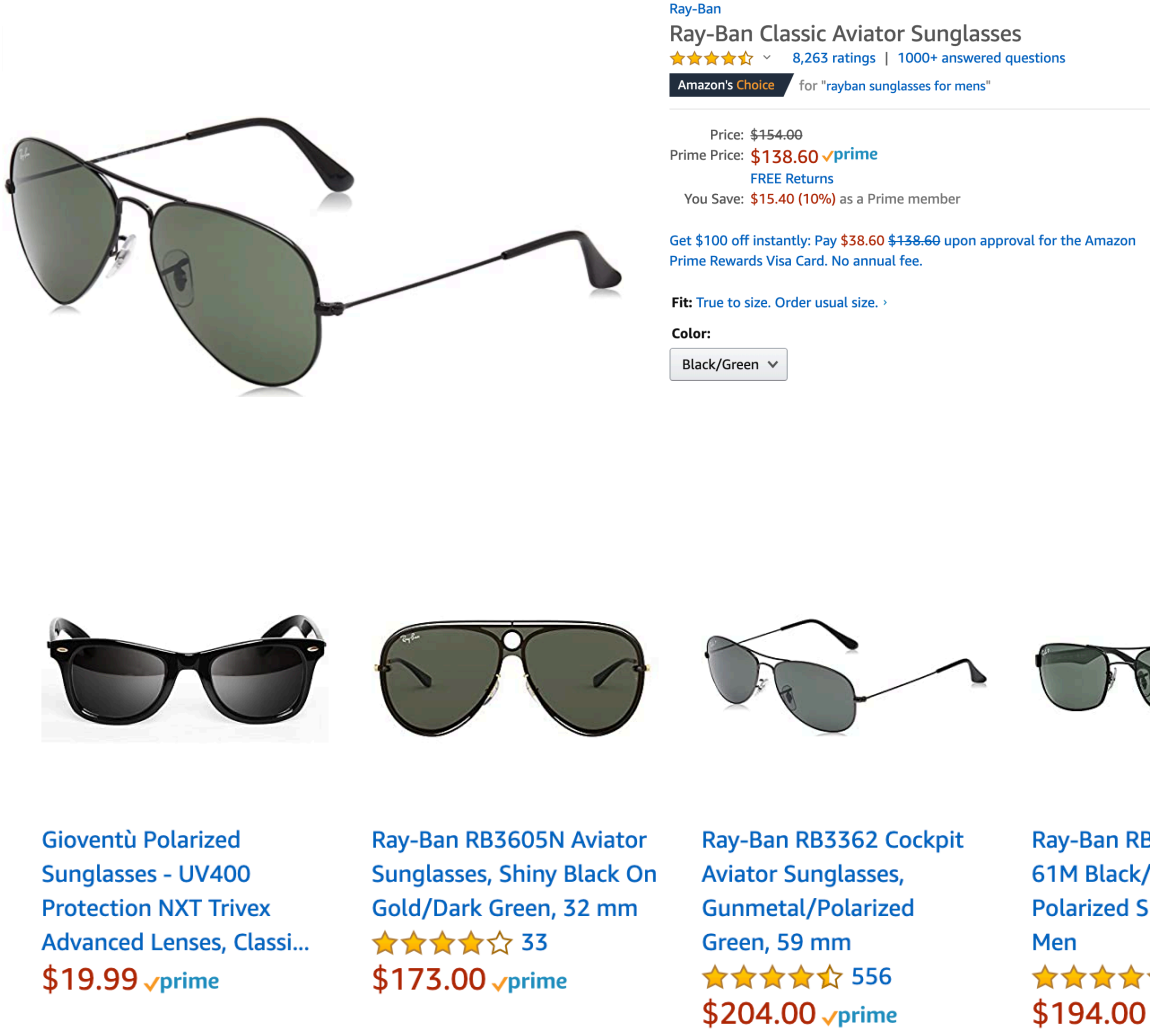
Under Prof. Hong Man.....

Index

Introduction to Recommender Systems

Introduction to Recommender Systems

- Recommender Systems are AI algorithms that generally use Machine Learning to recommend consumers with items which are closest to their purchased, reviewed or viewed product.
- The goal of recommender systems is to suggest the most relevant and personalized products to the consumer keeping his/her needs as a priority but at the same time generating maximum profits.



Introduction to Recommender Systems

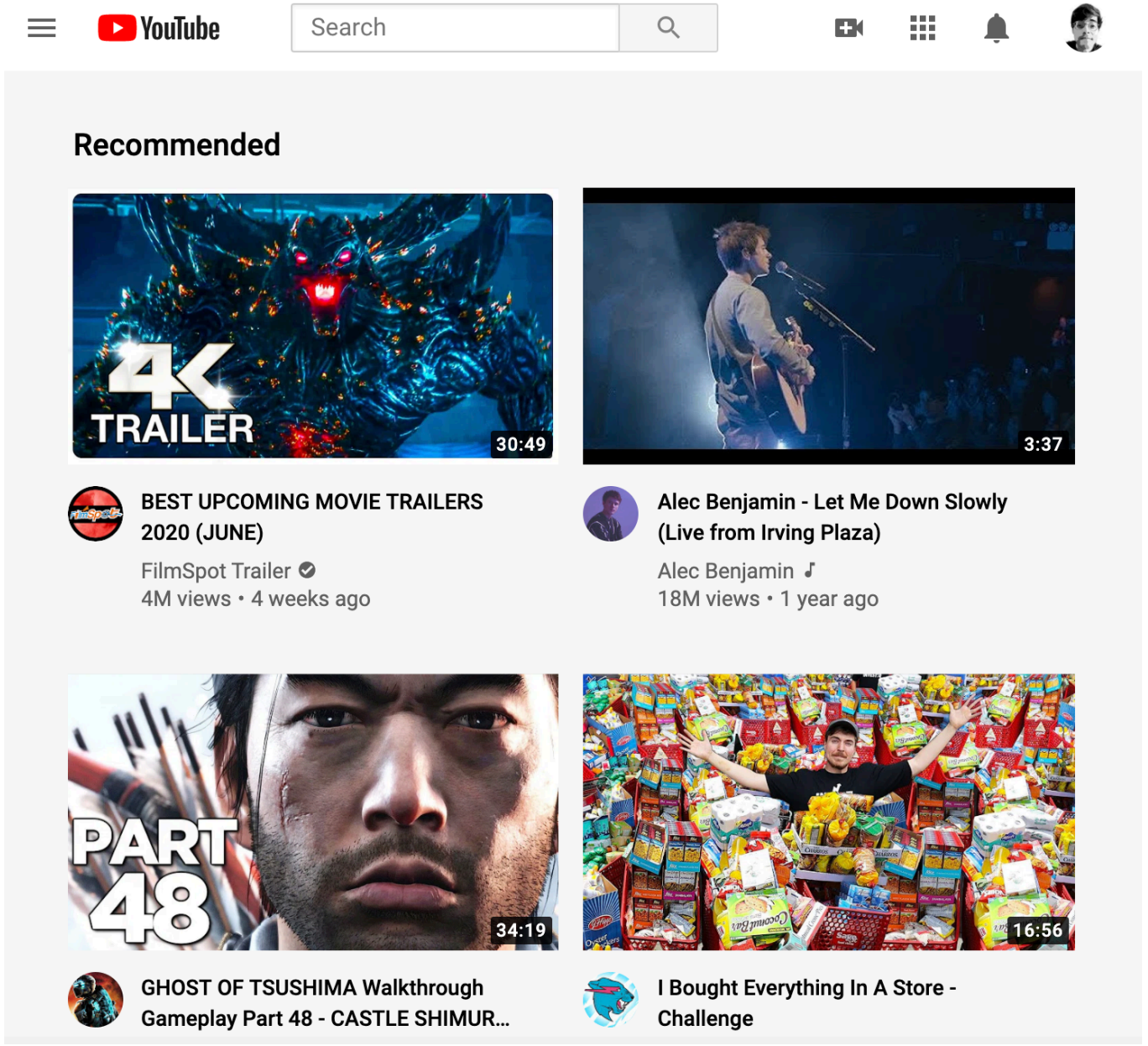
- In the media industry, giants such as Netflix, Spotify and YouTube recommend media based on millions of user's likes, views or comments on videos, music or movies or other features such as view timings, genre or type, name of the artists, actors or producers.
- The different kinds of Recommender Systems we will speak about in this research are as follows: –

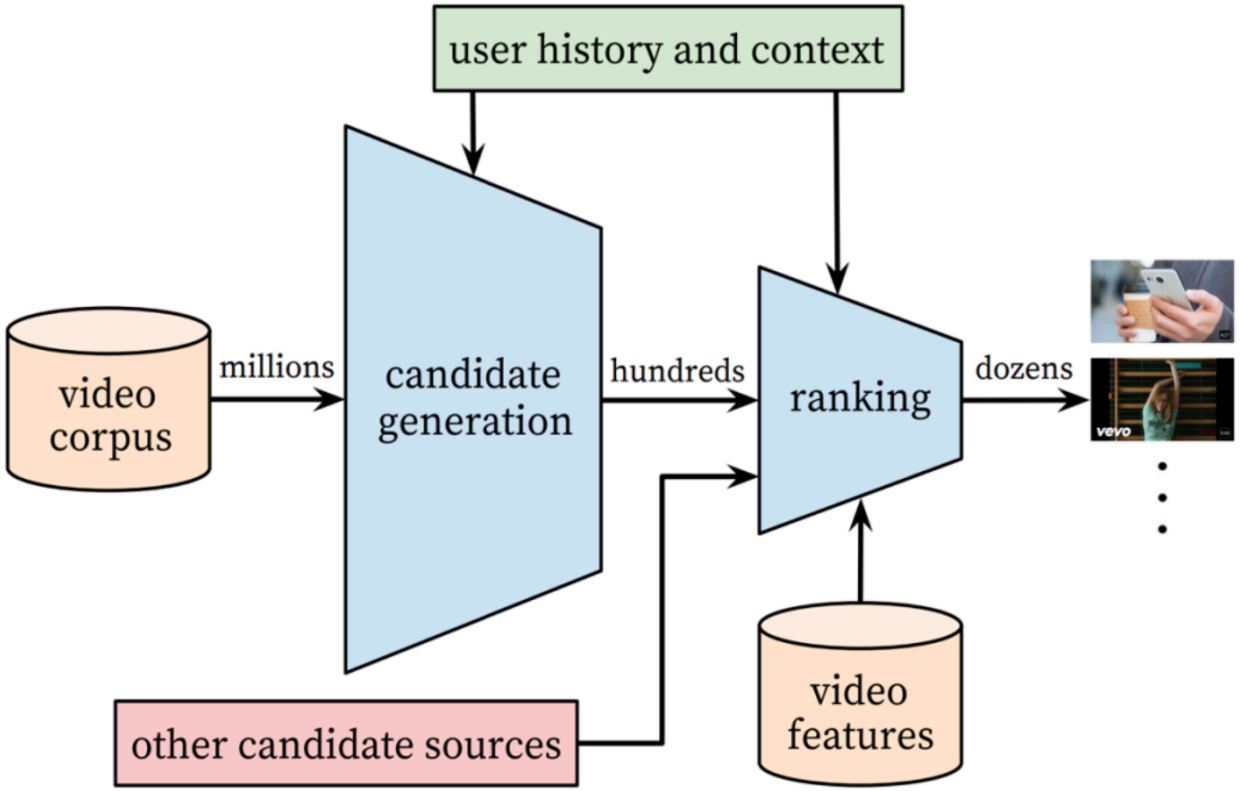
1. Content Based Filtering
2. Collaborative Filtering
3. Weighted Averages
4. ALS (Alternating Least Squares) based on PySpark

Case Studies of Recommender Systems in Media

Case Studies of Recommender Systems in Media

1. YouTube



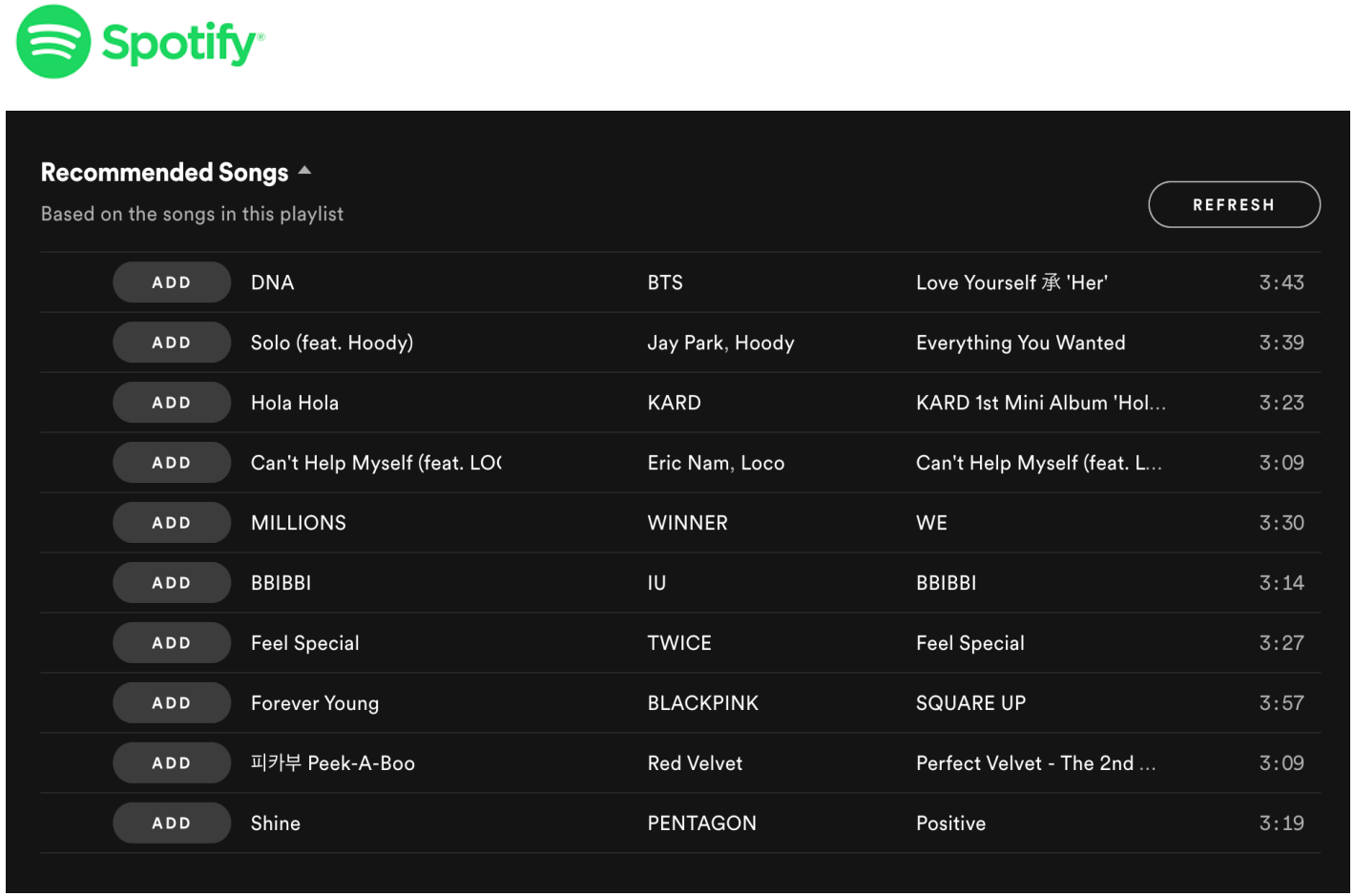


Case Studies of Recommender Systems in Media

- YouTube has a Two Stage Process for Recommending videos to its users.
- Candidate Generation and Ranking.
- Candidate Generation Algorithm takes millions of videos available on the platform and filters out the videos that the user may like.
- Ranking Algorithm sorts the videos selected by the former algorithm and sorts them in order of relevance.

Case Studies of Recommender Systems in Media

2. Spotify



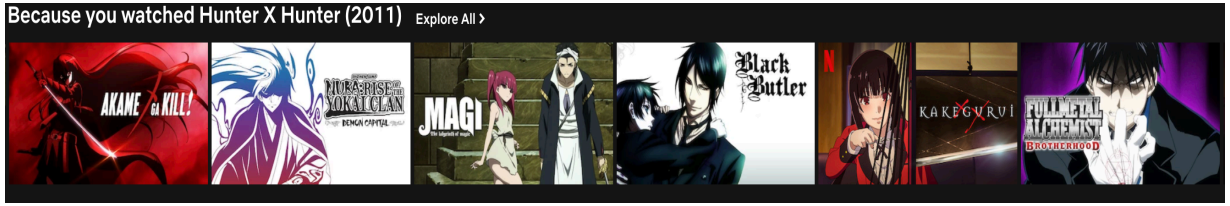
Case Studies of Recommender Systems in Media

- There are three recommendation models at work on Spotify:

1. Collaborative Filtering: This model uses "nearest neighbors" to make predictions about what other users might enjoy.
2. Natural Language Processing: – Spotify identifies the co-location of individual terms and uses this to predict the meaning of phrases.
3. Audio Models: – Spotify uses the kind of neural network that is employed by search engines to understand the contents of images. These networks process raw audio to produce a range of characteristics, including key, tempo, and even loudness. We have explored these models to recommend tracks based on our Spotify Models.

Case Studies of Recommender Systems in Media

3. Netflix



Case Studies of Recommender Systems in Media

- Netflix estimates the likelihood using Collaborative Filtering that the user will watch a movie in their catalog based on number of factors including:

1. The user interactions with the platform (such as your viewing history and how you rated other titles).
2. Other users with similar tastes and preferences on our service.
3. Information about the titles, such as their genre, categories, actors.
4. Year of release.

Netflix adds a lot of personalization in their recommender systems to produce top notch results which have completely disrupted the way TV and media industry operates.

Dataset Description

Dataset Description

- For our research we have used several datasets for training our models and validation they are listed below

1. Amazon Movies Dataset.
2. IMDB Reviews Dataset.
3. Spotify Playlist with audio features (spotipy API).

Dataset Description

1. AMAZON MOVIES DATSASET

	reviewerID	Movie_ID	reviewerName	helpful	reviewText	ratings	summary	unixReviewTime	reviewTime
0	ADZPIG9QOCDG5	0005019281	Alice L. Larson "alice-loves-books"	[0, 0]	This is a charming version of the classic Dick...	4	good version of a classic	1203984000	02 26, 2008
1	A35947ZP82G7JH	0005019281	Amarah Strack	[0, 0]	It was good but not as emotionally moving as t...	3	Good but not as moving	1388361600	12 30, 2013
2	A3UORV8A9D5L2E	0005019281	Amazon Customer	[0, 0]	Don't get me wrong, Winkler is a wonderful	3	Winkler's Performance	1388361600	12 30, 2013

Dataset Description

- The most important features we will be using during our research from this dataset are: -

1. reviewerID - This feature signifies the ID of every user.
2. Movie_ID - This feature signifies the ID of every movie.
3. reviewText - The comment left behind by every user.
4. ratings - Ratings provided by user. (On a scale of 1-5).
5. Sentiments - Our custom variable that signifies the sentiment of every comment (Positive, Negative and Neutral).

Dataset Description

2. IMDB DATASET

- This dataset contains 5000 movie reviews left by users.
- We use this dataset to generate our ratings and sentiment predictions.

	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 ...
4	Petter Mattei's "Love in the Time of Money" is...
...	...

Dataset Description

3. Spotify

- This dataset is imported using the spotipy API using the Spotify Developer Account.

	artist	album	track_name	track_id	danceability	energy	key	loudness	mode	speechiness	instrumentalness	livene
0	Bazzi	COSMIC	Mine	7uzmGiiJyRfuViKKK3lVmR	0.710	0.789	4	-3.874	1	0.0722	0.000003	0.451(
1	ZAYN	Mind Of Mine (Deluxe Edition)	PILLOWTALK	0PDUDa38GO8lMxLCRc4lL1	0.584	0.700	11	-4.275	1	0.0456	0.000000	0.093
2	Alex Aiono	Work The Middle	Work The Middle	42rB0s3mv8BpSKVBVkULvY	0.826	0.582	10	-7.162	0	0.2010	0.000005	0.147(
3	Drake	More Life	Passionfruit	5mCRDVBb16L4XQwDdbRUpz	0.809	0.463	11	-11.377	1	0.0396	0.085000	0.109(

Dataset Description

- The audio features are as follows:-

1. 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.
2. Energy: Energy is a measure from 0.0 to 1.0 and rep- resents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
3. 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.

Dataset Description

4. Liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.
5. 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.
6. 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.

Dataset Description

7. 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.
8. 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).

Project Flow

Project Flow

- Data Preprocessing: Every approach had a different set of preprocessing 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 pre- dictions on our test datasets. We have also generated music recommendations by training Spotify playlists and movie recommendations on IMDB.

Approaches to Recommender Systems

Content Based Filtering with Logistic Regression

- Content-based filtering uses item features to recommend similar items for users based on their previous actions or explicit feedback. 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',
```

Content Based Filtering with Logistic Regression

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

```
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
```

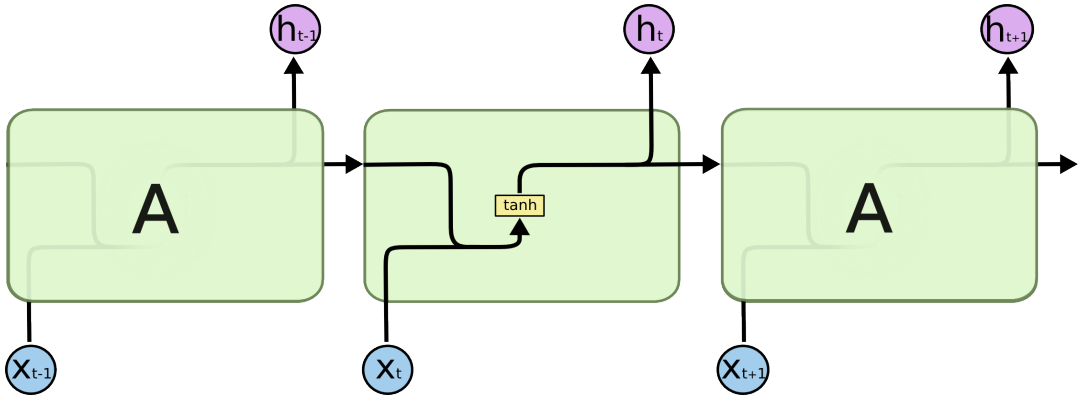
```
pipe = Pipeline([('cleaner', predictors()),
                 ('tfidfVect', tfdifVect),
                ('classifier',classifier),])
```

Content Based Filtering with LSTM

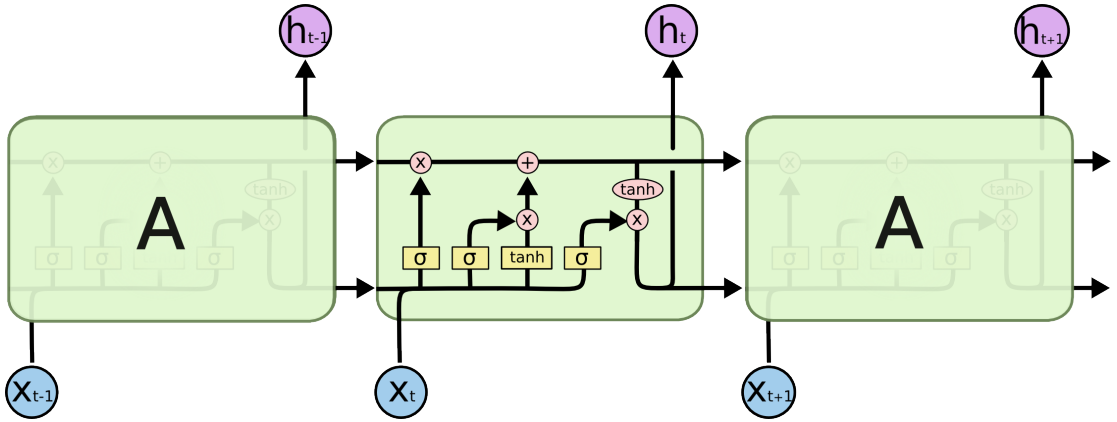
- Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. LSTMS are required for complex problem domains like machine translation, speech recognition, and more.
- All recurrent neural networks have the form of 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.
- We will be using LSTMs for generating content-based recommendations.

Content Based Filtering with LSTM

- General Architecture for RNN



- General Architecture for LSTM



Content Based Filtering with LSTM

- The data preprocessing for this approach is similar to Content Based Filtering with Logistic Regression approach.
- We split the reviews into bag of words while removing punctuations and stop words and count the number of times the words have repeated using our own custom CountVectorizer. 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
```

Content Based Filtering with LSTM

- 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.
- The training dataset is then fed to LSTM model which has a sigmoid activation function, cross entropy loss function and adam optimizer to generate recommendations.

```
X_train[0]
```

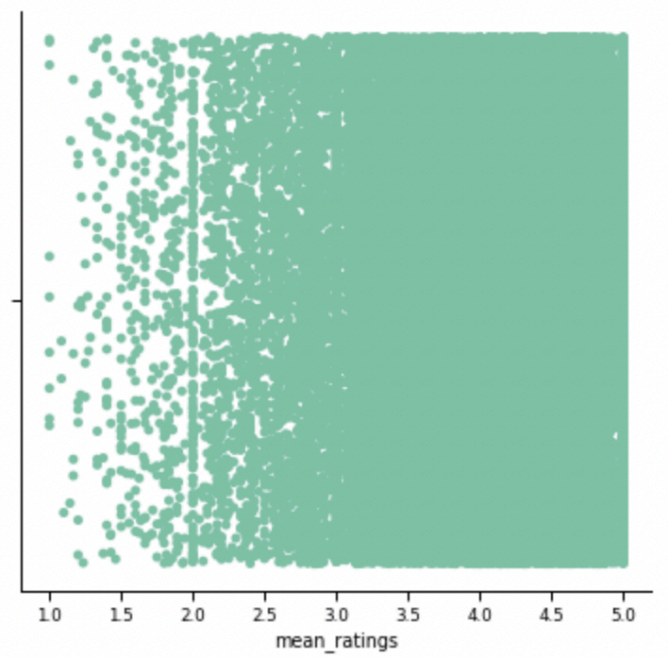
```
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,   574,     4,    55,     3,
        1431,    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])
```

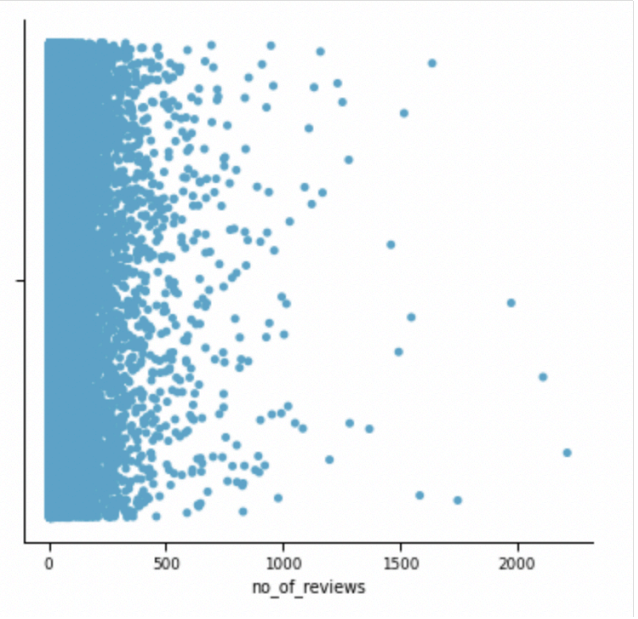
```
y_train[0]
```

```
1
length: 7999
```

Collaborative Filtering with KNN

- Collaborative filtering uses algorithms to filter data from user reviews to make personalized recommendations for users with the most similar preferences. There are several types of Collaborative Filtering, in this research we will be focusing on Neighbor based Filtering.
- In neighbor-based filtering, users are selected for their similarity to the active user. This similarity is determined by matching users who have posted similar reviews.
- For Data Preprocessing, we 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.





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	
...	...	...	

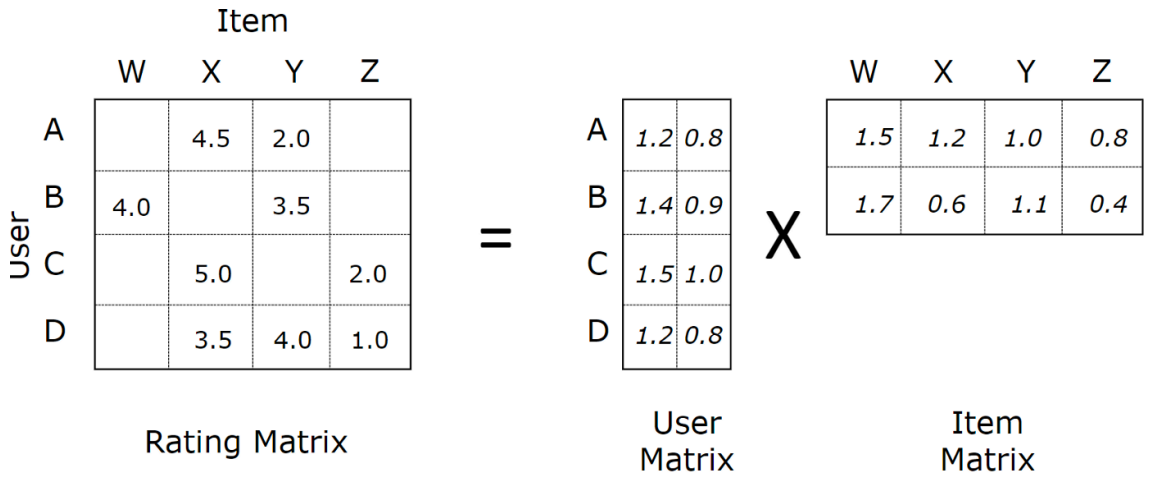
Collaborative Filtering with KNN

- From EDA and previous 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 reviews for a movie fall, they will not be considered.
- We then create 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.

Collaborative Filtering with KNN

- Pivot table occupies a lot of space when it comes to training a model hence, we use sparse matrix which only contains nonzero ratings. We use a scikit-learn based KNN model to train our training dataset and generate recommendations.
- For this approach we have also generated rating recommendations for the IMDB reviews dataset.

Collaborative Filtering with ALS



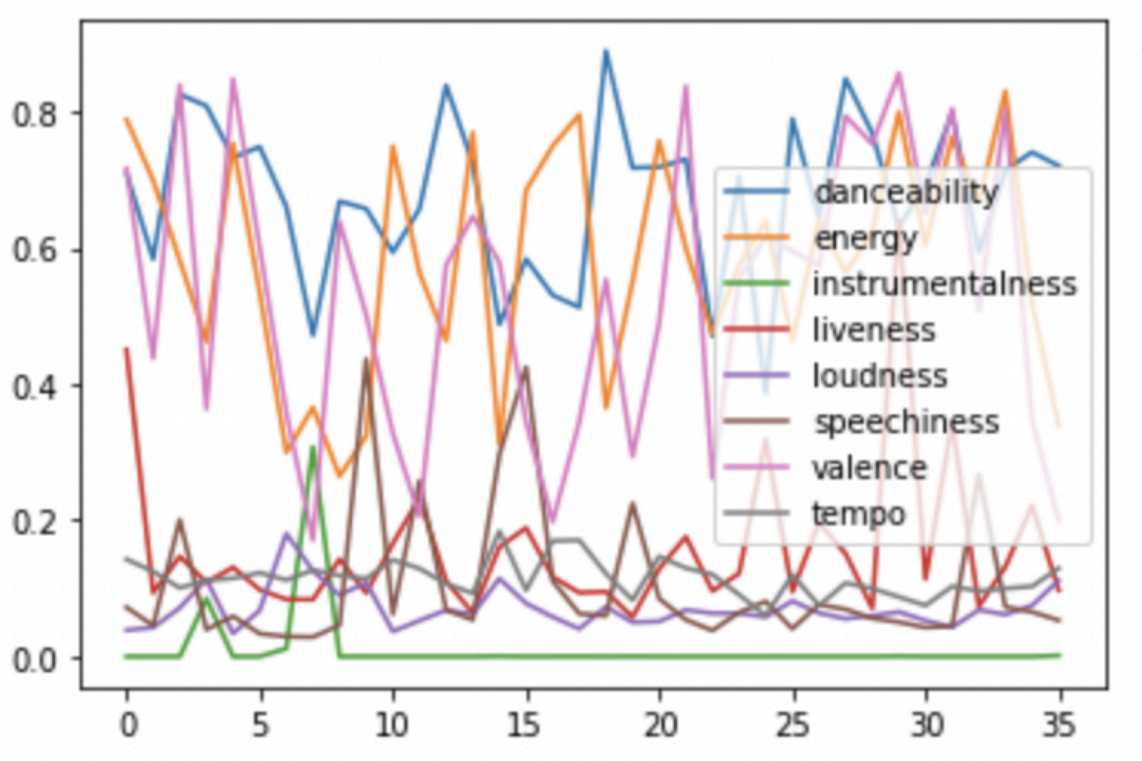
- ALS or the Alternating Least Square Model a matrix factorization algorithm and it runs itself in a parallel fashion. ALS is implemented in Apache Spark ML and built for a larges-scale collaborative filtering problems.
- ALS is doing a pretty good job at solving scalability and sparseness of the Ratings data, and it's simple and scales well to very large datasets.
- In our approach we have used the ALS model in PySpark to generate Recommendations.

Collaborative Filtering with ALS

- We begin by first creating a spark session and allocating memory size
- We then read the csv into a Spark DataFrame.
- 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.
- These features are then used to generate predictions and in turn generate recommendations using RegressionEvaluator and Pyspark based ALS model.

reviewerID	Movie_ID	ratings
ADZPIG9QOCDG5	0005019281	4
A35947ZP82G7JH	0005019281	3
A3UORV8A9D5L2E	0005019281	3
A1VKW06X1O2X7V	0005019281	5
A3R27T4HADWFFJ	0005019281	4
A2L0G56BNOTX6S	0005019281	5

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



Spotify Recommender Engine

- We have already spoken about the different Spotify audio features in our dataset description.
- Rescaling these features we can compare and analyze them and the user's preferences for a playlist.

Spotify Recommender Engine

- We create a custom function based on the spotipy API to import our playlists from Spotify along with its audio features (such as loudness, valence, acousticity, etc.) along with artist name, album name and song name.
- These features are rescaled and hypertuned to understand what kind of music does the user prefer.
- 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.

Results

Results

- For the Content Based Filtering

1. Our Logistic Regression our model yielded a Validation Accuracy of 0.8526 and we have applied this model to generate recommendations for IMDB reviews.
2. LSTM on the other hand yielded an accuracy of 0.769.

	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

Results

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

```
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
```

Results

- ALS based PySpark model has lowest computational time and is by far the fastest model to train because of parallel computing.

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

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...

Results

- 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
```

Results

- IMDB Weighted Average Algorithm was also used along with collaborative filtering to generate recommendations based on max weighted average ratings.
- The algorithm → $W = \frac{Rv + Cm}{v + m}$

	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
...	...	...	...

Conclusion and Future Scope

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 personalized 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.

Future Scope

- Content Creator's Perspective – A further point of research could be to 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.
- Image Processing – Images give a lot of information of the data. For example movie posters can be processed and categorized into a genre and new movies could be recommended based on these genres.

Thank You