Analysis of IMDb data

Sfurti Gaudani, Sasyak Pattnaik, Atmika Sarukkai

Introduction

<u>2 main research questions</u>:

1) What can we learn about differences in words in different genres? Which genres have more unique words? What words are most common for a given genre?
    a) Publicists who are drafting overviews can find this model useful
    b) Successful Overview → popularity, viewership, revenue

2) How can we determine optimal (elements) for different features, based on past performance indicators?
    a) Use public opinion, critique reviews, and overall popularity to find best actor per genre
    b) Recommendation system

Data & Pre-processing

IMDB Dataset

- Includes top 1000 movies obtained from Kaggle
- Granularity: one row per movie

Poster_Link	Series_Title	Released_Ye	Certificate	Runtime	Genre	IMDB_Rating	Overview	Meta_score	Director	Star1	Star2	Star3	Star4	No_of_Votes	Gross
https://m.m	The Shawsha	1994	A	142 min	Drama	9.3	Two impriso	80	Frank Darabo	Tim Robbins	Morgan Free	Bob Gunton	William Sadl	2343110	28,341,469
https://m.m	The Godfath	1972	A	175 min	Crime, Dram	9.2	An organizec	100	Francis Ford	Marlon Bran	Al Pacino	James Caan	Diane Keator	1620367	134,966,411
https://m.m	The Dark Kni	2008	UA	152 min	Action, Crime	9	When the m	84	Christopher I	Christian Bal	Heath Ledge	Aaron Eckha	Michael Cain	2303232	534,858,444

Columns :

- Link
- Name
- Year released
- Certificate
- Runtime
- Genre
- IMDB Rating (public rating)
- Overview
- Meta Score (critic rating)
- Director
- Top 4 Stars
- # of Votes
- Gross (earnings)

Data Pre-processing

One-Hot Encoding:

Helps categorize which movie had a certain quality

Genre example:

- One movie can have multiple genres
- Movie will have a 1 in the "Horror" column if it falls into the "Horror" genre

Horror	Music	Musical	Mystery	Romance	Sci-Fi	Sport	Thriller	War	Western
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

Data Pre-processing

Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre
https://m.media-amazon.com/images/M/MV5BMDFkYT...	The Shawshank Redemption	1994	A	142 min	['Drama']
https://m.media-amazon.com/images/M/MV5BM2MyNj...	The Godfather	1972	A	175 min	['Crime', 'Drama']

Step 1: Change genre from string to a list of strings

Poster_Link	Series_Title	Released_Year	Certificate	Runtime	Genre	IMDB_Rating
https://m.media-amazon.com/images/M/MV5BMTMxNT...	The Dark Knight	2008	UA	152 min	Action	9.0
https://m.media-amazon.com/images/M/MV5BNzA5ZD...	The Lord of the Rings: The Return of the King	2003	U	201 min	Action	8.9

Step 2: Construct df such that a movie appears once for each genre it falls into

Data Pre-processing

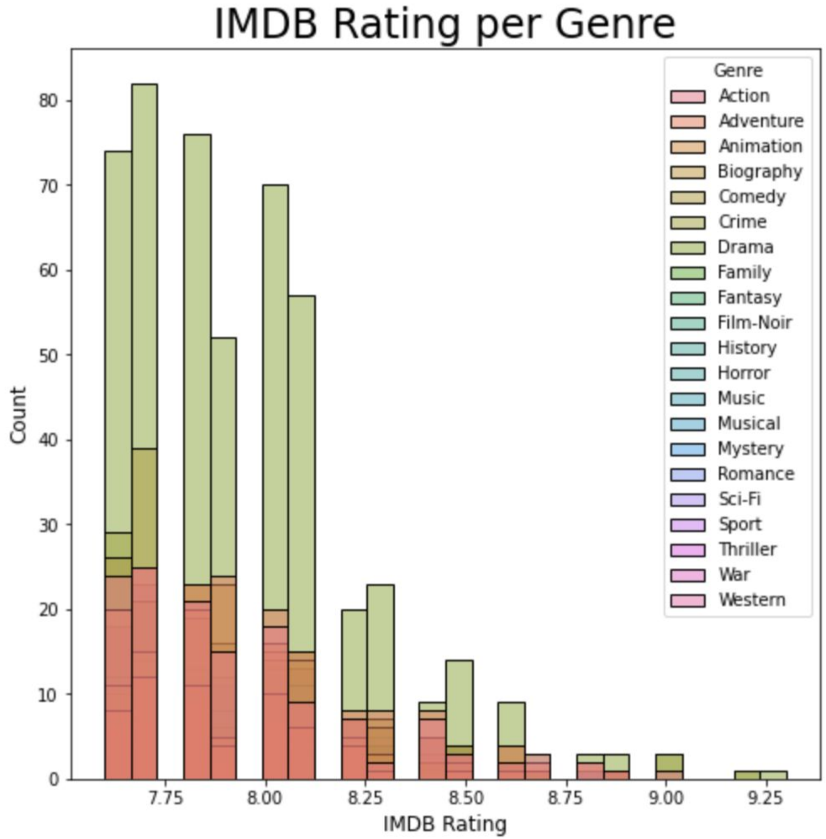
Data Cleaning (for graphs):

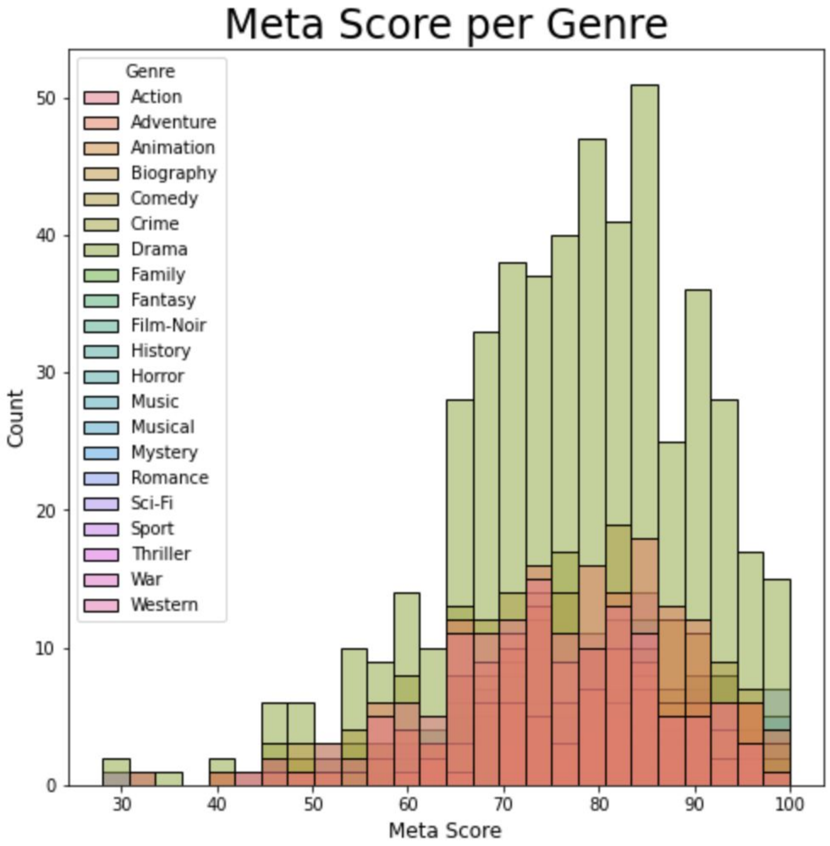
- Ignored NA values when creating visualizations
- Used str attributes to clean and convert “Gross” and “Runtime” to int

```
df_new["Runtime"] = df_new["Runtime"].str.replace(" min", "")
df_new["Runtime"] = df_new["Runtime"].astype(int)
avg_runtime = df_new.groupby("Genre")["Runtime"].mean()
avg_runtime = avg_runtime.to_frame().reset_index()
```

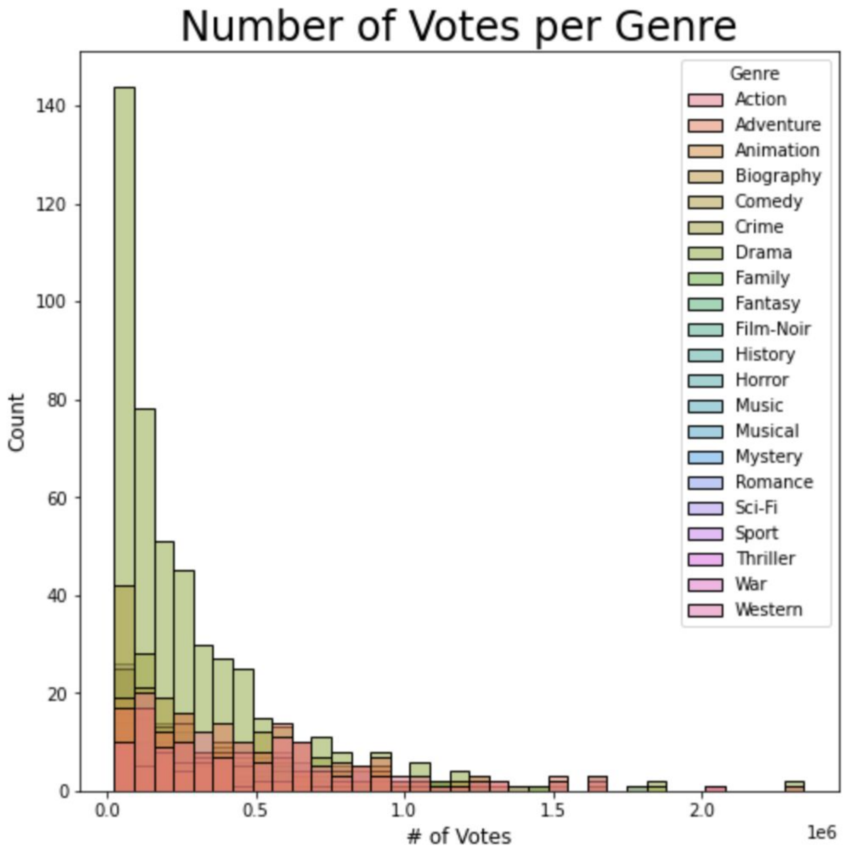
Visualizations & EDA

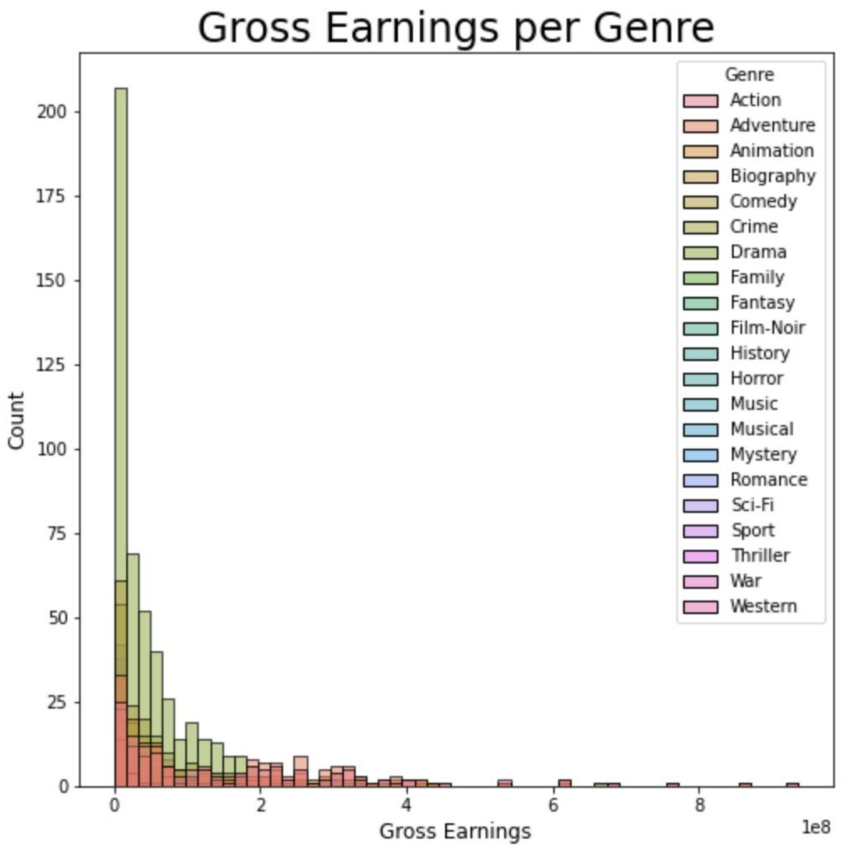
Visualizations/EDA



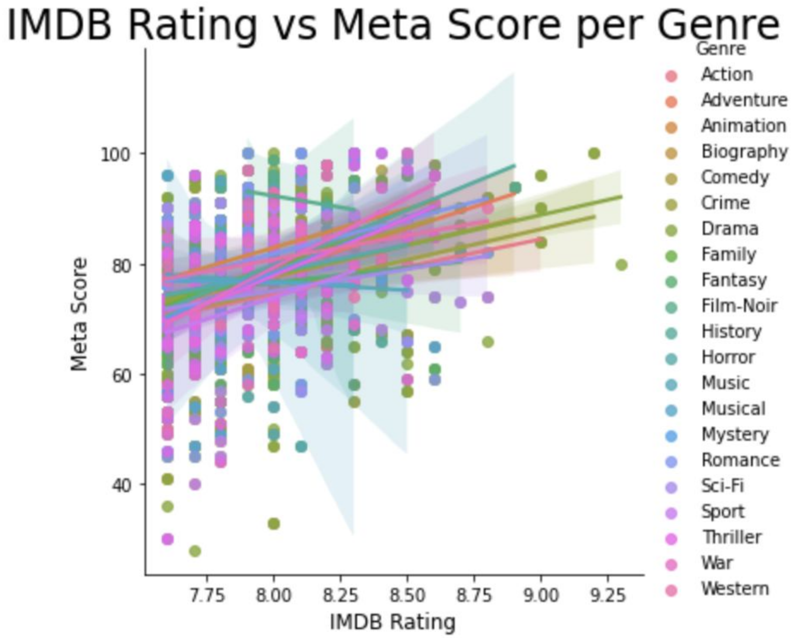


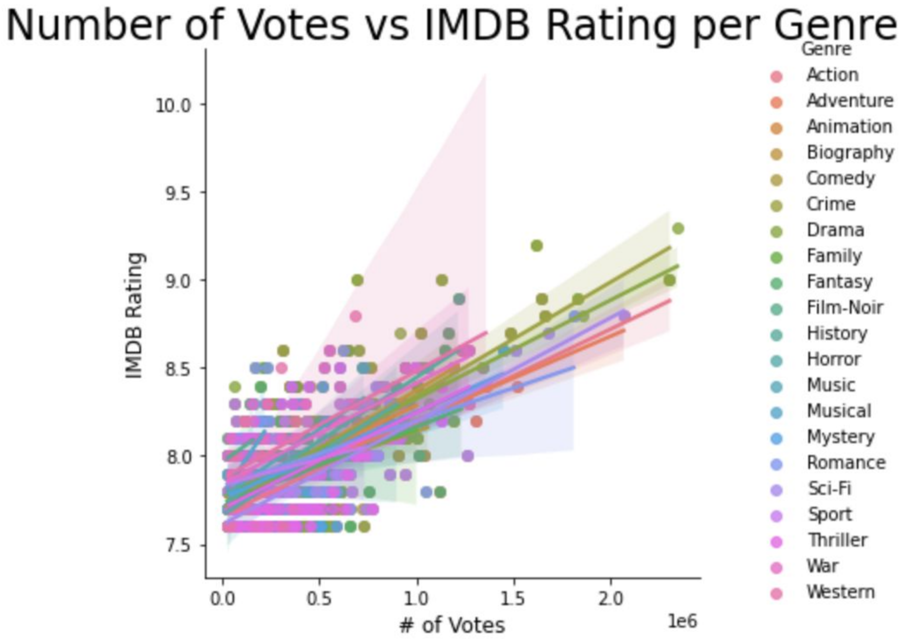
Visualizations/EDA



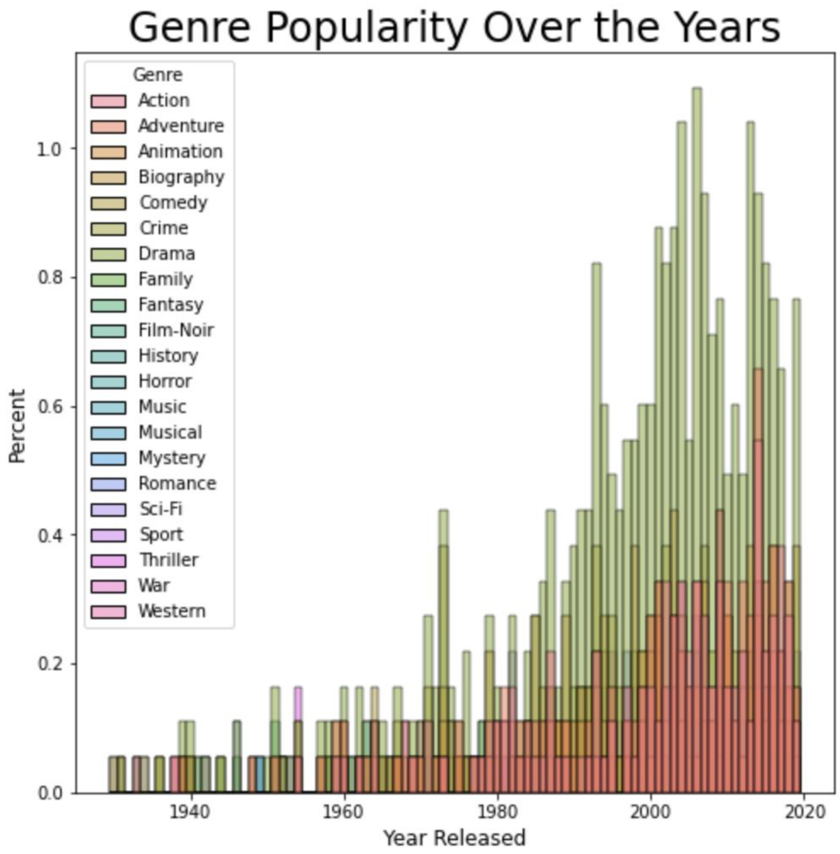


Visualizations/EDA

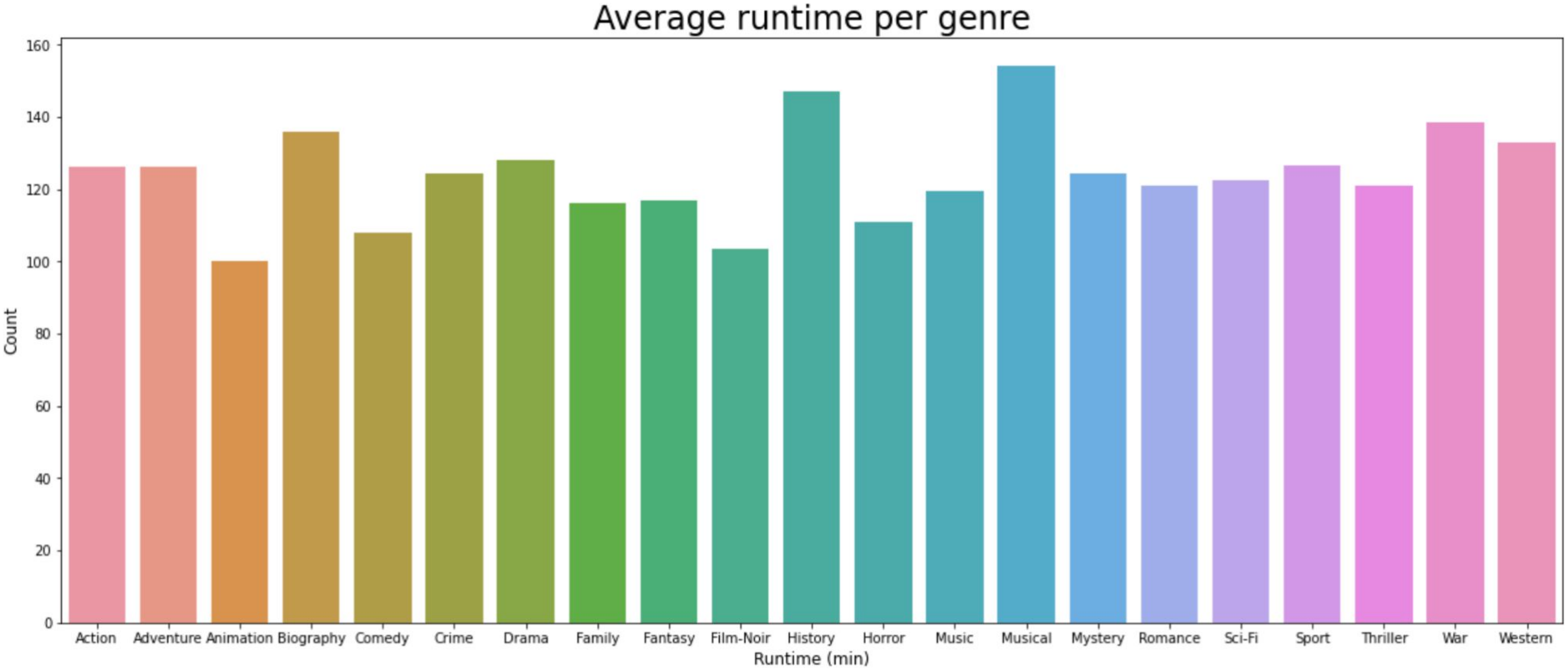




Visualizations/EDA



Visualizations/EDA



Natural Language Processing

Word Embeddings

- “Overview” column
- Regex expression → tokenization
- Remove punctuation
- Lower case all unique words
- Input unique words into **Word2Vec** model → vector representations of each word

```
coal [-0.00102609  0.01473691  0.0052536  -0.00548682  0.01342138 -0.0169
766
 -0.00272759  0.01315213 -0.00545542 -0.00083087  0.00469587 -0.00139453
 -0.01072146  0.00754463  0.01135934 -0.01723334  0.00250256 -0.00238994
 -0.00137512 -0.01161674 -0.00073253 -0.00095731  0.01661374  0.00050033
 -0.00410702  0.00732082  0.00416431  0.01006746 -0.00854229 -0.0039268
  0.00499693  0.00111887 -0.00014614 -0.00542992 -0.00512275 -0.00433421
  0.0059452  -0.0043379   0.00070274 -0.00115023 -0.00424259 -0.00984682
  0.0034081  -0.00021377  0.00429738  0.00754318 -0.01367367 -0.01382209
  0.00781415  0.00364194  0.00867755 -0.01118215  0.00300181 -0.0069665
 -0.00915049  0.01076606 -0.00297765 -0.00925238 -0.00141307  0.01059764
  0.00714071 -0.00417424  0.01338254  0.00704202  0.00196483  0.00342844
  0.00373915  0.00196996 -0.00220613  0.00182021 -0.00997894 -0.00361037
  0.00446524 -0.00355907  0.00746963 -0.0047117   0.00872123 -0.00438533
 -0.00185447  0.00965273 -0.00328426 -0.00346572 -0.0012068  -0.00134118
 -0.00908536 -0.01158435  0.01485254 -0.00449365  0.00071676  0.00275118
  0.0147505  -0.00476275 -0.00189954 -0.00530735  0.0102464   0.01517421
  0.01404284 -0.01362117  0.00115778  0.01066232]
```

Vector representation of “coal”

Genre for unique words

- Remove stopwords from list of tokenized words
- For each word: count # movies in each genre the word is present in
- Identify genre with maximum # instances → can be ties

competition	border	exceptional	boy's	something	maps	high-class	bicycle	stumbles	tickets
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

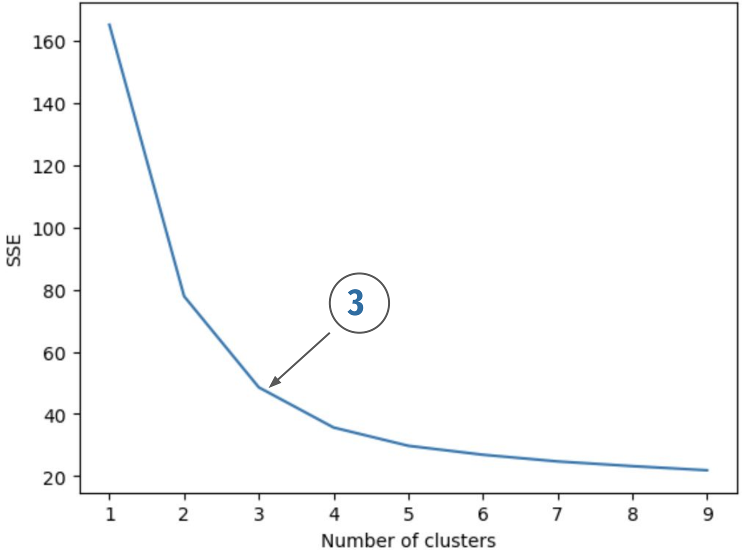
0: word is not in movie

1: word is in movie

Multiple genres per word

	word	word genre
0	showed	[Adventure, Biography, Drama]
1	coal	[Biography, Drama, Family]
2	student	[Drama]
3	woman	[Drama]
4	profile	[Biography, Drama, Music]
...	...	...
5458	riley	[Adventure, Animation, Comedy]
5459	soapmaker	[Drama]
5460	journalist	[Drama]
5461	elevate	[Drama]
5462	diagnosed	[Drama]

K means clustering

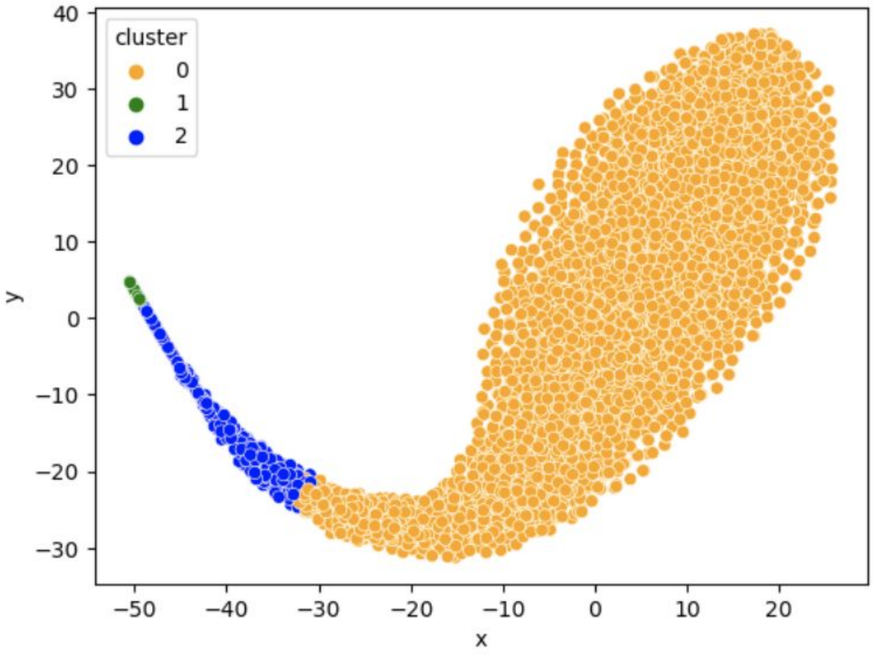


Elbow method

Word embedding

	x	y	word	cluster	word genre
0	0.324411	-5.979132	showed	0	[Adventure, Biography, Drama]
1	5.633975	-9.395093	coal	0	[Biography, Drama, Family]
2	-47.485790	-10.596395	student	1	[Drama]
3	-47.967075	-16.783575	woman	2	[Drama]
4	19.567457	-7.124603	profile	0	[Biography, Drama, Music]
...	...	...	...	...	...
5458	18.600473	-21.035786	riley	0	[Adventure, Animation, Comedy]
5459	0.142635	2.486060	soapmaker	0	[Drama]
5460	-40.440910	7.476940	journalist	1	[Drama]
5461	17.140202	-10.573872	elevate	0	[Drama]
5462	-12.982013	24.637785	diagnosed	0	[Drama]

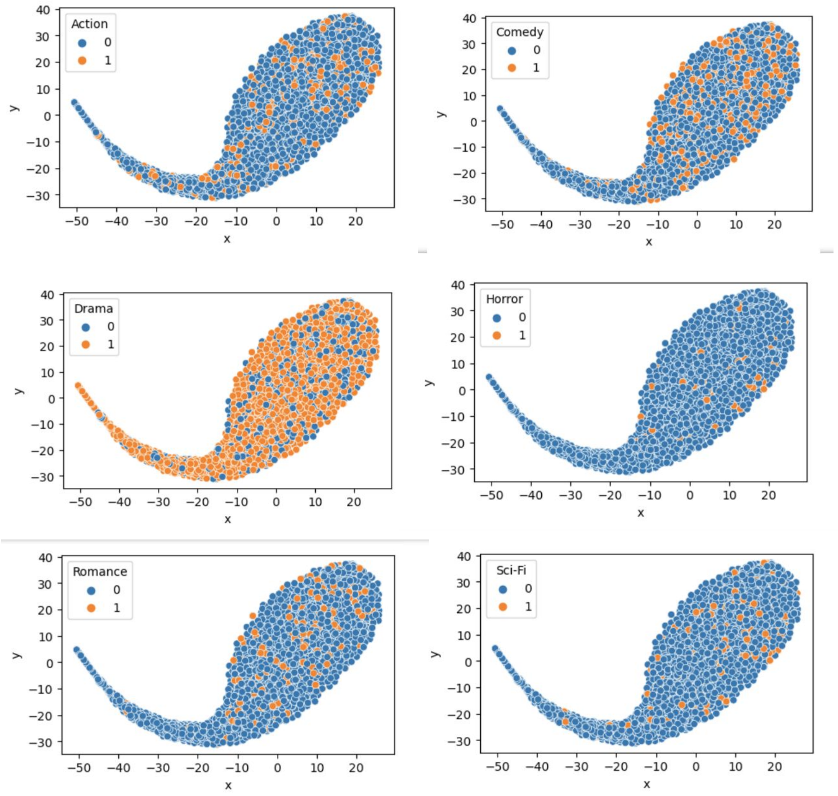
Dimensionality Reduction (T-SNE)



	cluster_0	cluster_1	cluster_2
Drama	3534	50	411
Comedy	1091	0	41
Adventure	1060	0	60
Action	926	0	43
Crime	856	1	29
Thriller	572	0	6
Biography	567	0	4
Romance	467	0	2
Animation	399	0	2
Mystery	374	0	5
Sci-Fi	359	0	18
Fantasy	303	0	3
Family	271	0	3
History	268	0	1
War	194	0	3
Horror	153	0	2
Music	148	0	0
Sport	91	0	0
Musical	73	0	1
Film-Noir	73	0	1
Western	62	0	5

* k -means clusters do not separate genres

Visualizing word embeddings for genre



	cluster_0	cluster_1	distances
Action	(-3.0062063, -0.1024115)	(1.4078835, 2.3902154)	5.069258
Adventure	(-2.9609425, -0.014051254)	(0.63725096, 1.7115232)	3.990564
Animation	(-2.6698802, 0.015181669)	(3.4146702, 4.436488)	7.521283
Biography	(-2.9128132, -0.011932336)	(3.6844678, 3.3524594)	7.405623
Comedy	(-3.186216, -0.09340183)	(1.460999, 1.996824)	5.095650
Crime	(-3.2845988, -0.3482086)	(3.259545, 3.8934903)	7.798579
Drama	(0.34805837, 1.8218855)	(-3.1681106, -0.20491773)	4.058494
Family	(-2.5722091, 0.103034675)	(4.3851876, 4.822023)	8.406796
Fantasy	(-2.6050594, 0.12148908)	(4.211239, 4.017523)	7.851178
Film-Noir	(-2.2456994, 0.37494305)	(-0.5888726, -2.2254949)	3.083399
History	(-2.539957, 0.17306788)	(3.8917694, 3.557497)	7.267838
Horror	(-2.414669, 0.23249717)	(4.3316827, 4.0115256)	7.732679
Music	(-2.4310086, 0.13968131)	(5.2375546, 7.5234804)	10.645532
Musical	(-2.3250256, 0.2848841)	(5.1879807, 4.3329906)	8.534192
Mystery	(-2.6951427, 0.017288484)	(4.1067405, 4.6648726)	8.238061
Romance	(-2.9384685, -0.22004785)	(5.3924513, 6.300214)	10.579132
Sci-Fi	(-2.5615795, 0.19019254)	(2.3409579, 2.3569283)	5.360002
Sport	(-2.3656802, 0.21845232)	(6.1844287, 7.4984117)	11.229522
Thriller	(-2.8495605, 0.054627076)	(3.069983, 2.749184)	6.503970
War	(-2.4129033, 0.2427827)	(2.846179, 2.9309022)	5.906262
Western	(-2.2754858, 0.29193518)	(1.9831247, 4.1880517)	5.771957

Music, Romance, Sport

^ clusters farthest apart

Word Identifier Function

Genre / List of Genres

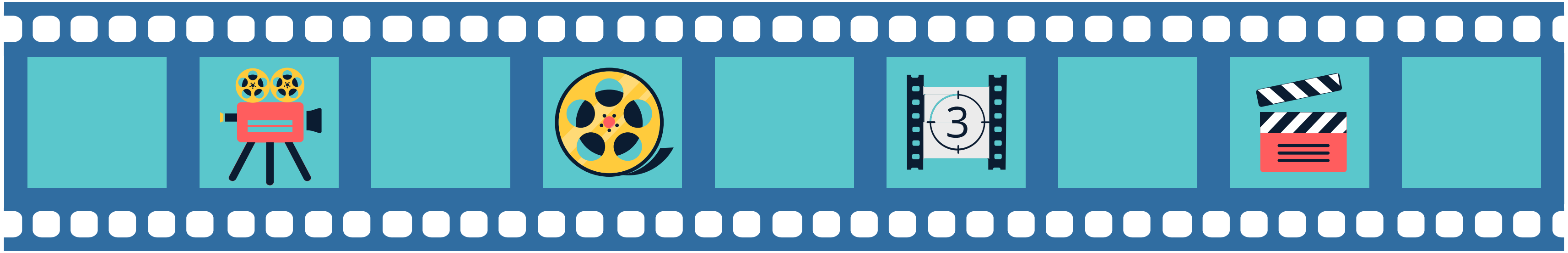
Ex: [Music, Romance]

List of Words

Ex: Words most closely associated with Music **and** Romance

```
array(['busker', 'disguised', '1924', 'setting', 'male', 'liner',
       'discovered', 'coming-of-age', 'downward', 'peking', 'send',
       'elise', 'all-female', 'didier', 'pursuit', 'modern-day',
       'alcoholism', 'ballerina', 'prima', 'witness', 'hit', 'musical',
       '1900', 'rehearse', 'spite', 'ocean', 'rappers', 'spiral',
       'ballet'], dtype='<U13')
```

Additional Functions and Recommendations



Best Actor ~ Genre

Goal: To analyse historical data points and determine the best proven actor for any selected genre.

Technique: Using a scoring metric to rate individual actor performances based on IMDB public ratings, critique Meta ratings, and the number of overall voters which is representative of the actor’s popularity.

Scoring System: *IMDB Rating(~25%) + Meta Rating(~25%) + Total Votes(~50%)*

Conclusion: We were able to implement a function that uses the above scoring metric in analysing the historic preprocessed data we collected to determine individual best actors per genre, and we estimate that the accuracy of this function is fairly accurate based on the results we collected(displayed next).

Model Results


```
[ ] genre_to_best_actor("Drama")

    'Robert De Niro'
```


```
[ ] genre_to_best_actor("Romance")

    'Ethan Hawke'
```

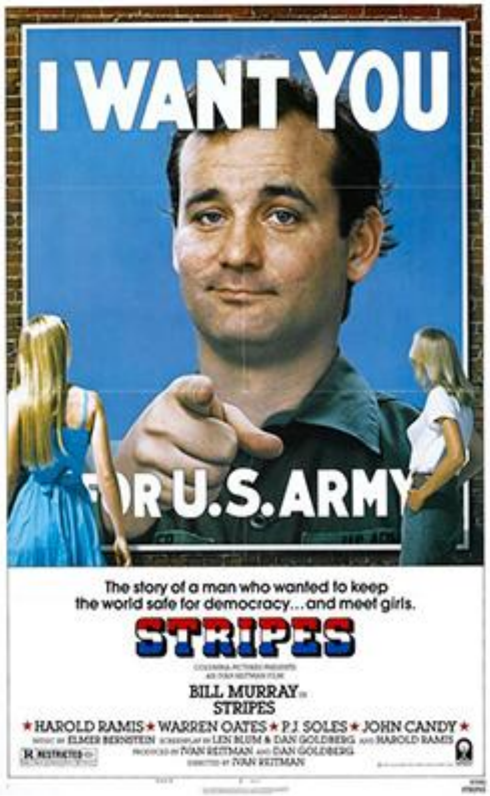
```
[ ] genre_to_best_actor("Adventure")

    'Ian McKellen'
```

Model Results

```
[ ]  genre_to_best_actor("Action")

     'Harrison Ford'
```




```
[ ]  genre_to_best_actor("Comedy")

     'Bill Murray'
```

Recommender

Recommender System

Goal: To get the top few movie recommendations based on user inputted movie

Techniques: Vectorization, Dimension Reduction, Cosine Similarity

Result: We were able to use the above techniques to vectorize and truncate our data into a format that could be used to determine similarities between different movies. The features we used for comparing movies included the genres, the overviews of the movie and the primary movie actors. We then used the Cosine similarity method to calculate distances between our determined vectors, and thereafter select the top 10 most similar movies(results on next page).

Methods Used:

1) **TfidVectorizer:** This is an alternative method for CountVectorizer, and measures the originality of a word in a dataframe.
2) **TruncatedSVD:** This is used for dimensionality reduction, which simplifies the model by reducing the number of input variables which then increases the performance of the model.
3) **Cosine_similarity:** This is a measure used to determine the similarity between two inputted variables based on the cosine angle between the two vectors. It produces a value between 0 and 1, which indicates the similarity between the vectors.

Model Recommendations

```
recommendation('The Lord of the Rings: The Two Towers')

Avengers: Endgame
Avengers: Endgame
Avengers: Infinity War
Avengers: Infinity War
Avengers: Infinity War
Captain America: Civil War
The Avengers
Captain America: Civil War
Captain America: Civil War
```

```
recommendation('The Dark Knight')

Inception
Inception
Wo hu cang long
Sicario
Waking Life
Waking Life
Waking Life
The Odd Couple
Gandhi
```

```
recommendation('The Godfather')

The Lord of the Rings: The Return of the King
The Lord of the Rings: The Return of the King
The Lord of the Rings: The Fellowship of the Ring
The Lord of the Rings: The Fellowship of the Ring
The Lord of the Rings: The Fellowship of the Ring
The Lord of the Rings: The Two Towers
The Lord of the Rings: The Two Towers
The Lord of the Rings: The Two Towers
X-Men: Days of Future Past
```

```
recommendation('Inception')

Shichinin no samurai
Shichinin no samurai
The Last Samurai
The Last Samurai
The Magnificent Seven
The Magnificent Seven
The Magnificent Seven
Skyfall
Seven Pounds
```

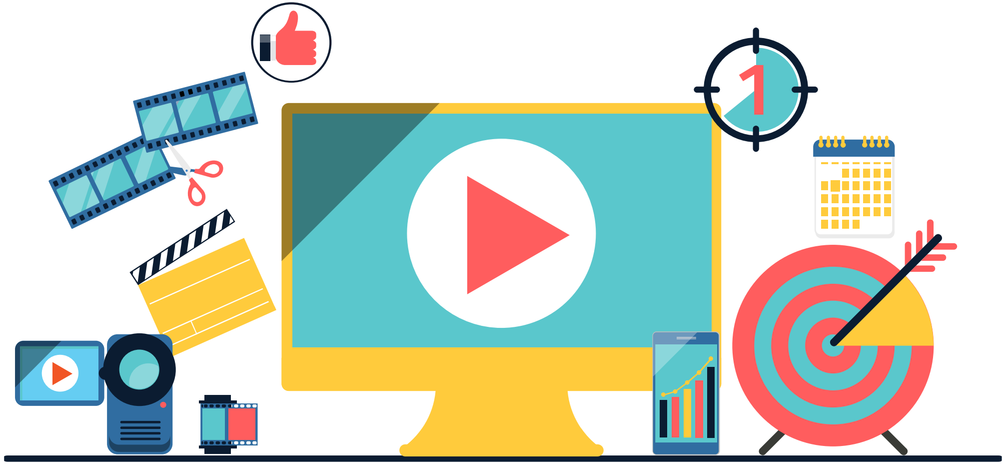
```
recommendation('The Magnificent Seven')

Million Dollar Baby
Nebraska
Nebraska
Nebraska
Arrival
The Sound of Music
Arrival
Scarface
Todo sobre mi madre
```

```
recommendation('True Grit')

Lion
Home Alone
Capharnaüm
Home Alone
The Lion King
Oldeuboi
The Goonies
The Lion King
The Lion King
```

Results + Conclusions



Results + Conclusions

NLP

- Findings: From word embeddings, we learn that certain genres (such as Music, Romance, and Sport) have more unique words
- Implications:
  - May be useful for publicists to decide what key words to include in overview of movie plot
  - May be useful for aspiring directors/writers to understand what are key characteristics of a genre
  - May be beneficial for movie popularity/revenue if certain words associated with genre

Best Actor Function

- Findings: best actor given a genre
- Implications
  - May help in movie casting

Recommender

- Findings: identified a list of similar movies given an inputted movie
- Implications:
  - a user can successfully generate a list of movies based on their liking of individual movies of their liking
  - allows users to easily navigate among movies of their liking and avoid having to browse through a random list of movies to find what they are looking for

Results + Conclusions

Our findings are not ready to be used in the real-world :

- Limitations of data
  - Only 1000 movies
  - Contains movies from as early as the 1940s – might not be as helpful for current new movies/suggestions
- Limitations of Best Actor function
  - Function doesn’t consider the cost of hiring top actors
  - Recommender doesn’t consider specific roles that an actor might play (roles that are age, gender, race specific)

What needs to be solved before it is ready to be used :

- Larger corpus of data of top movies
- Data on movies in more recent years
- Data on characteristics of actors and their availability to take on roles

^ for better understanding relationships between genres & improving recommender system

Thank you!