COGS 118B - Final Project

Movie Recommendation System through Clustering Techniques

Group members

- Pranjli Khanna
- Vishnudeep Tyagi
- Hongjo Yoon
- Arturo Torres

Contributions:

- Pranjli Khanna: Fixing the abstract based on feedback, worked on preprocessing data, worked on evaluating data, evaluations
- Vishnudeep Tyagi: Fixing Proposed solution, problem statement, abstract, discussion, results (one section)
- Hongjo Yoon: Fixing the data based on feedback, worked on recommendation model, Fixed problem statement based on feedback, limitations, subsection 1, worked on abstract, fixing proposed solution
- Arturo Torres: Problem Statement on proposal

Abstract

This project addresses the need for a movie recommendation system by creating a system that suggests movies to users based on their interests. By utilizing an open-source dataset from Kaggle, which includes detailed movie metadata from IMDb, including information such as movie titles, genres, and overviews, we preprocessed the data for analysis while maintaining user privacy. Our approach combined Natural Language Processing (NLP) and clustering, particularly K-Means, to organize movies into similar groups based on their synopses. Dimensionality reduction was applied using Principal Component Analysis (PCA) and Latent Dirichlet Allocation (LDA) to streamline our data and uncover underlying topics.

Our key achievement is the development of a recommendation function that, given a user's preference keyword, identifies a corresponding movie cluster and recommends the top movies within that cluster. The recommendations are based on cosine similarity scores, ensuring relevance and accuracy. This project demonstrates how clustering techniques and NLP can be leveraged to enhance user experiences in digital content platforms by providing personalized movie suggestions.

Background

In an era dominated by digital streaming, the landscape of television and movie consumption has dramatically transformed, leading to an exponential increase in the volume of content available to viewers. This transformation has brought to the forefront the challenge of content discovery, making it increasingly difficult for viewers to navigate through endless options and select what to watch next. The inception of this project is motivated by the desire to tackle the growing dilemma of choice overload, aiming to develop a sophisticated movie recommendation system that harnesses the power of machine learning to offer personalized content suggestions.

The traditional approach to watching television, characterized by fixed schedules and limited selections, has been upended by the advent of streaming services, offering a boundless array of movies and TV shows at the fingertips of consumers. This shift has not only redefined viewer habits but has also introduced the problem of decision fatigue, where the abundance of choices paradoxically hampers the viewer's ability to make a selection. The project seeks to address this issue by creating a recommendation system that simplifies the decision-making process, enhancing the overall viewing experience.

According to a report highlighted in the context of streaming media consumption, the average American spends approximately 45 hours a year deciding what to watch next[1]. This statistic underscores the significant amount of time that could be better spent engaging with content rather than sifting through it. The project's relevance is thus underscored by this finding, as it aims to reduce the time spent in content selection, leveraging algorithms to streamline the process and make it more efficient. Further emphasizing the shift in viewer preferences, a report also points to the rise of on-demand and mobile viewing, indicating a clear departure from traditional TV screen viewing[2]. This trend not only reflects changing technological landscapes but also a broader desire for flexibility and personalization in media consumption. By aligning the project's objectives with these trends, the proposed recommendation system seeks to cater to the modern viewer's needs, offering tailored suggestions that resonate with individual preferences and viewing habits.

The demand for solutions capable of addressing these challenges is also reflected in the job market, with a notable increase in listings for recommender systems positions[3]. This demand signals a growing recognition of the value and necessity of advanced recommendation systems within the industry. By tapping into this need, the project not only aligns with current technological and market trends but also contributes to an area of significant interest and investment.

In conclusion, the project is poised at the nexus of technological innovation and evolving viewer habits, seeking to address the pressing challenge of content discovery in the streaming era. By drawing on insights from consumer behavior, industry trends, and the evident demand for skilled professionals in recommender systems, the project underscores the urgency and relevance of developing a solution that enhances content accessibility and personalization. Through the development of an advanced movie recommendation system, this project aims to redefine how viewers interact with streaming content, making the process more intuitive, enjoyable, and aligned with individual preferences.

Problem Statement

The problem solved by our movie recommendation model is the amount of time wasted by people in the process of trying to find a movie to watch. Whether it be a problem of arguing with others over what to watch for the night, or simply trying to expand the type of media that you consume, it can be difficult to find something in particular when you have no idea what to watch, only a vague idea of what might be entertaining. This is a problem due to the fact that, throughout your life, wasting precious time on something such as being stuck on what movie to watch, the time wasted tends to add up. Our solution to this problem is to leverage clustering techniques to develop a model that can recommend movies based on simple one-word prompts. By streamlining the selection process through personalized recommendations, we aim to alleviate the time burden associated with movie selection and enhance the overall viewing experience for users.

Data

The dataset we will be using in our project will be information regarding movies that users have given ratings for. These ratings will range from 0 to 100 and were collected anonymously. This dataset was given from IMDb and contains variables such as the name of the movie, genre, release date, crew members, and most importantly the description of the movie (labeled under “overview” in the dataset). This dataset contains 12 variables some of which were aforementioned although some variables will be dropped due to their lesser importance to our model such as release status and original language. As for the observations there seems to be over 10,000 different movies which are represented in this dataset. In order to utilize this dataset we first had to clean the dataset and drop the columns that we thought were not useful enough to include in our function such as the date it was released, the language it was originally released in, and the budget of the movie. The most important part of the data, the ‘overview’ was preprocessed by us in order to make sure that the words were usable. The way that we preprocessed the data was turning every word into lowercase, removing stop words such as ‘an’ ‘if’ ‘or’, and turning words into lemma where they drop the ‘ing’ part of the word. After which we put the preprocessed ‘overview’ into a new column called ‘preprocessed_synopsis’. After that in order to actually use the words in a meaningful way we tokenized them using TF-IDF to turn the words into numerical representations. The cleaning process we took was located in a file called ‘preprocess.ipynb’ and below is the data before we processed it

	names	date_x	score	genre	overview	crew	orig_title	status	orig_lang	budget_x	revenue	country
0	Creed III	03/02/2023	73.0	Drama, Action	After dominating the boxing world, Adonis Cree...	Michael B. Jordan, Adonis Creed, Tessa Thompso...	Creed III	Released	English	75000000.0	2.716167e+08	AU
1	Avatar: The Way of Water	12/15/2022	78.0	Science Fiction, Adventure, Action	Set more than a decade after the events of the...	Sam Worthington, Jake Sully, Zoe Saldaña, Neyt...	Avatar: The Way of Water	Released	English	460000000.0	2.316795e+09	AU
2	The Super Mario Bros. Movie	04/05/2023	76.0	Animation, Adventure, Family, Fantasy, Comedy	While working underground to fix a water main,...	Chris Pratt, Mario (voice), Anya Taylor-Joy, P...	The Super Mario Bros. Movie	Released	English	100000000.0	7.244590e+08	AU
3	Mummies	01/05/2023	70.0	Animation, Comedy, Family, Adventure, Fantasy	Through a series of unfortunate events, three ...	Óscar Barberán, Thut (voice), Ana Esther Albor...	Momias	Released	Spanish, Castilian	12300000.0	3.420000e+07	AU
4	Supercell	03/17/2023	61.0	Action	Good-hearted teenager William always lived in ...	Skeet Ulrich, Roy Cameron, Anne Heche, Dr Quin...	Supercell	Released	English	77000000.0	3.409420e+08	US

And underneath that is what the data looked like after we had processed it.

	names	score	genre	overview	crew	preprocessed_synopsis	preprocessed_genre	tokenized_crew
0	Creed III	73.0	Drama, Action	After dominating the boxing world, Adonis Cree...	Michael B. Jordan, Adonis Creed, Tessa Thompso...	dominate box world adonis creed thrive career ...	drama action	[Michael B. Jordan, Adonis Creed, Tessa Thomps...
1	Avatar: The Way of Water	78.0	Science Fiction, Adventure, Action	Set more than a decade after the events of the...	Sam Worthington, Jake Sully, Zoe Saldaña, Neyt...	set decade event film learn story sully family...	science fiction adventure action	[Sam Worthington, Jake Sully, Zoe Saldaña, Ney...
2	The Super Mario Bros. Movie	76.0	Animation, Adventure, Family, Fantasy, Comedy	While working underground to fix a water main,...	Chris Pratt, Mario (voice), Anya Taylor-Joy, P...	work underground fix water main brooklyn plumb...	animation adventure family fantasy ...	[Chris Pratt, Mario (voice), Anya Taylor-Joy, ...
3	Mummies	70.0	Animation, Comedy, Family, Adventure, Fantasy	Through a series of unfortunate events, three ...	Óscar Barberán, Thut (voice), Ana Esther Albor...	series unfortunate event mummy end present day...	animation comedy family adventure ...	[Óscar Barberán, Thut (voice), Ana Esther Albo...
4	Supercell	61.0	Action	Good-hearted teenager William always lived in ...	Skeet Ulrich, Roy Cameron, Anne Heche, Dr Quin...	good hearted teenager william live hope follow...	action	[Skeet Ulrich, Roy Cameron, Anne Heche, Dr Qui...

Proposed Solution

The solution that we came up with was using the preprocessed tokenized data that we had to describe the movies. We used KMeans in order to cluster the similar movies together by their TFIDF representations in order to see which cluster our input word would most likely belong to. We also used PCA (Principal Component Analysis) in order to reduce the dimensionality of the TFIDF vectors so that our recommendations were to be as accurate as possible. Lastly we used LDA (Latent Dirichlet Allocation) to again reduce dimensionality so that we could make sure that our synopsis data could be better separated and better categorized. Our function would take the string input from the user which would then be preprocessed in the same way that our data from the dataset was, vectorize it using TFIDF, then try to predict which cluster the user input word belonged to, which would calculate the cosine similarity score between the movies and the input word and would then output the top five movies with the best similarity scores. Important libraries we used included, sklearn, pandas, and spacy. And the functions were named preprocess(text) which would process the text to be usable, and recommend_movies(keyword) which the keyword would be the user input of a string which would return the list of movies. The way in which we will test our solution will be through the evaluation functions that we created in order to see the Precision@5 and Recall@5 scores in order to see if our recommendations were precise.

Evaluation Metrics

Propose at least one evaluation metric that can be used to quantify the performance of both the benchmark model and the solution model. The evaluation metric(s) you propose should be appropriate given the context of the data, the problem statement, and the intended solution. Describe how the evaluation metric(s) are derived and provide an example of their mathematical representations (if applicable). Complex evaluation metrics should be clearly defined and quantifiable (can be expressed in mathematical or logical terms).

In evaluating the performance of our movie recommendation system, it is essential to select metrics that accurately reflect the quality and relevance of the recommendations provided to users. Given the context of our data and the problem statement, which involves clustering and recommending movies based on textual similarity, Precision@k and Recall@k emerge as suitable evaluation metrics. These metrics offer a balance between the relevance of the recommendations and the system's ability to retrieve all relevant items, which are critical for assessing a recommendation system's effectiveness.

Precision@k

Precision@k measures the proportion of recommended items in the top-k set that are relevant. It is defined as the number of relevant recommendations divided by k, the number of recommendations made. This metric is particularly useful in scenarios where the goal is to maximize the number of relevant recommendations presented to the user, while limiting the number of irrelevant items. The mathematical representation of Precision@k is given by:
Precision@k = Number of Relevant Items in Top-k Recommendations / k

Recall@k

Recall@k, on the other hand, measures the proportion of relevant items that are successfully retrieved in the top-k recommendations. It is defined as the number of relevant recommendations divided by the total number of relevant items in the dataset. This metric assesses the system's ability to capture as many relevant items as possible. The mathematical representation of Recall@k is:
Recall@k = Number of Relevant Items in Top-k Recommendations / Total Number of Relevant Items

Example

Consider a scenario where a user searches for "basketball" related movies, and the recommendation system suggests five movies. Among these, two are known to be relevant based on a ground truth set of relevant basketball movies. Assuming our ground truth contains five relevant basketball movies, the Precision@5 and Recall@5 can be calculated as follows:

- Precision@5: The system recommended five movies, out of which two were relevant. Therefore, Precision@5 is 2/5= 0.4 or 40%.
- Recall@5: Out of the total five relevant basketball movies in the ground truth, the system correctly recommended two. Therefore, Recall@5 is 2/5 = 0.4 or 40%.

These metrics are essential for understanding the trade-offs between the relevance of recommendations (Precision@k) and the system's comprehensiveness in retrieving relevant items (Recall@k). By optimizing these metrics, we can enhance the performance of the movie recommendation system, ensuring users receive the most pertinent recommendations based on their interests.

	names	score	genre	overview
2798	Space Pirate Captain Harlock	66.0	Animation, Science Fiction	Space Pirate Captain Harlock and his fearless ...
8828	Space Chimps	48.0	Animation, Family, Adventure, Comedy, Science ...	Three chimps are sent into space to explore th...
9940	High Life	58.0	Science Fiction, Drama, Mystery	A father and his daughter struggle to survive ...
6510	Lifeforce	63.0	Horror, Science Fiction, Thriller	A space shuttle mission investigating Halley's...
3719	Gattaca	75.0	Thriller, Science Fiction, Mystery, Romance	In a future society in the era of indefinite e...

```python
def evaluate_recommendation_system(query, ground_truth, recommend_function, k=5):
    """
    Evaluate the recommendation system for a given query using Precision@k and Recall@k.

    Parameters:
    - query: The query keyword used for generating recommendations.
    - ground_truth: A dictionary mapping queries to lists of relevant movie titles.
    - recommend_function: The function used to generate movie recommendations.
    - k: The number of top recommendations to consider for evaluation.

    Returns:
    - precision_k: Precision@k for the given query.
    - recall_k: Recall@k for the given query.
    """
    # Generate top-k recommendations for the query
    recommended_movies = recommend_function(query)
    recommended_titles = set(recommended_movies['names'].tolist())

    # Get the ground truth relevant movies for the query
    relevant_movies = set(ground_truth[query])

    # Calculate the number of relevant recommendations
    relevant_recommendations = recommended_titles.intersection(relevant_movies)
    num_relevant_recommendations = len(relevant_recommendations)

    # Calculate Precision@k and Recall@k
    precision_k = num_relevant_recommendations / k
    recall_k = num_relevant_recommendations / len(relevant_movies) if relevant_movies else 0

    return precision_k, recall_k

# Example usage:
ground_truth = {
    'space': ['Space Pirate Captain Harlock', 'Space Chimps', 'High Life', 'Lifeforce', 'Gattaca']  # Example ground truth
}

query = 'space'
precision_k, recall_k = evaluate_recommendation_system(query, ground_truth, recommend_movies, k=5)
print(f"Precision@5: {precision_k}")
print(f"Recall@5: {recall_k}")
```

✓ 0.0s

```
Precision@5: 1.0
Recall@5: 1.0
```

```python
def evaluate_recommendation_system(query, ground_truth, recommend_function, k=5):
    """
    Evaluate the recommendation system for a given query using Precision@k and Recall@k.

    Parameters:
    - query: The query keyword used for generating recommendations.
    - ground_truth: A dictionary mapping queries to lists of relevant movie titles.
    - recommend_function: The function used to generate movie recommendations.
    - k: The number of top recommendations to consider for evaluation.

    Returns:
    - precision_k: Precision@k for the given query.
    - recall_k: Recall@k for the given query.
    """
    # Generate top-k recommendations for the query
    recommended_movies = recommend_function(query)
    recommended_titles = set(recommended_movies['names'].tolist())

    # Get the ground truth relevant movies for the query
    relevant_movies = set(ground_truth[query])

    # Calculate the number of relevant recommendations
    relevant_recommendations = recommended_titles.intersection(relevant_movies)
    num_relevant_recommendations = len(relevant_recommendations)

    # Calculate Precision@k and Recall@k
    precision_k = num_relevant_recommendations / k
    recall_k = num_relevant_recommendations / len(relevant_movies) if relevant_movies else 0

    return precision_k, recall_k

# Example usage:
ground_truth = {
    # 'basketball': ['Above the Rim', 'The Way Back', 'He Got Game', 'Kuroko\'s Basketball the Movie: Last Game', 'Triple Standard']  # Example ground truth
    'space': ['Above the Rim', 'Mummies', 'Supercell', 'Kuroko\'s Basketball the Movie: Last Game', 'Gattaca']  # Example ground truth
}

query = 'space'
precision_k, recall_k = evaluate_recommendation_system(query, ground_truth, recommend_movies, k=5)
print(f"Precision@5: {precision_k}")
print(f"Recall@5: {recall_k}")
```

✓ 0.0s

```
Precision@5: 0.2
Recall@5: 0.2
```

Results

Through our experimentation and testing, we discovered that it is possible to create a movie recommendation system that takes the overview of the movie into account when a user inputs a descriptive word in order to gather movies that are similar or utilize that descriptive input in their movies. We have found this to be especially useful through the usage of KMeans combined with PCA and LDA to

be able to gather a small list of recommendations, which in turn will be able to reduce the amount of time that the average person wastes trying to think of a good movie to watch.

Something that we earlier tried to implement to achieve this was GMM (Gaussian Mixture Model) instead of LDA. GMM tended to not work too well with our data–perhaps because of the way that we formatted the data. As as a result, creating a function using GMM instead of LDA tended to not work and gave us very little variance in the recommendations.

Subsection 1

When we started to work with our data we looked at the different machine learning models and techniques that could be useful with our data and the problem that we set out to solve. We figured out that we needed to reduce dimensionality in order to make our data actually usable. Hence, we landed on using KMeans and PCA as mentioned above which helped reduce the dimensionality of the large dataset and the TFIDF vectors, which then led to us trying to find a specific model such as LDA which was able to find the similarities between the groups. Some techniques we decided not to use included linear regression, since our data wasn't something like numerical trend in which we could easily use linear regression, and we didn't use Manifold Learning as we didn't think that it would do a good enough job in clustering the data in the way that we wanted similar objects grouped together.

Subsection 2

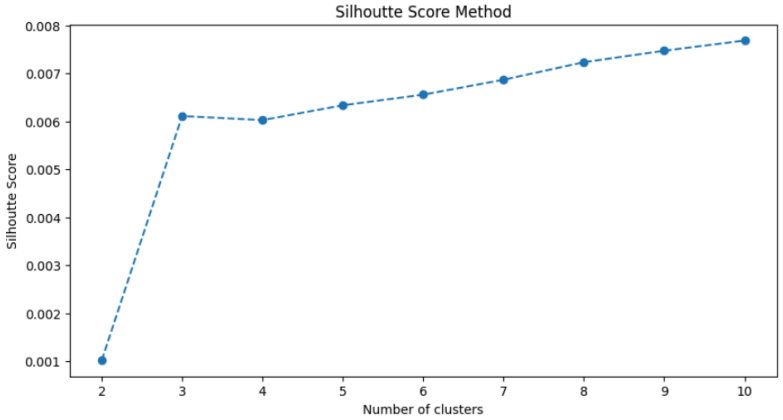
Another likely section is if you are doing any feature selection through cross-validation or hand-design/validation of features/transformations of the data

The feature selection of our data that we did was utilizing TFIDF of our preprocessed synopsis data. As seen here

```
# Tfidf object
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['preprocessed_synopsis'])
```

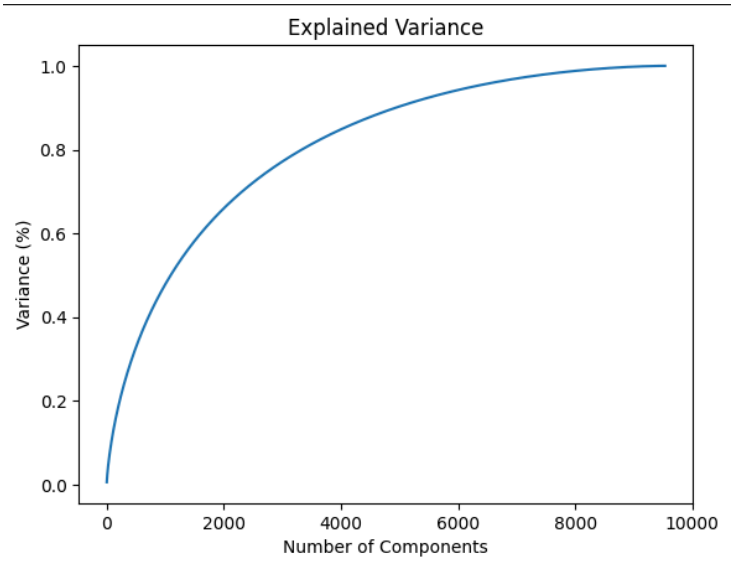
✓ 0.2s

We also used a silhouette score function in order to tune our hyperparameter for KMeans as seen in this code underneath the "Silhouette Score" section



Where we found that around 4 was the best number to use for the amount of clusters in KMeans. Additionally finding out the most efficient PCA n_components was figured out through creating a plot of

the explained variance vs the number of components which we found to be around 3000 which was in the "PCA Plot" portion of our code



Subsection 3

We explored the selection of hyperparameters and algorithms through validation curves and cross-validation. While our chosen approach yielded satisfactory results, we also experimented with alternative techniques such as Gaussian Mixture Model (GMM). However, GMM showed limited efficacy with our data, likely due to formatting issues, resulting in negligible variance in recommendations.

Subsection 4

Evaluation metrics such as precision and recall were employed to assess the effectiveness of our recommendation system. Precision@5 and recall@5 metrics were calculated for different query inputs, providing insights into the system's performance across various scenarios.

The results obtained from our movie recommendation system showcase its potential to significantly improve the movie selection experience for users. By harnessing machine learning algorithms such as KMeans, PCA, and LDA, we successfully reduced the dimensionality of the dataset and extracted meaningful features from movie overviews and genres. This enabled us to generate personalized recommendations based on user input, effectively addressing the inefficiencies associated with traditional movie selection methods. While our primary approach yielded promising results, our exploration into alternative techniques such as Gaussian Mixture Model (GMM) highlighted the importance of model selection and the need for careful consideration of data compatibility. Moving forward, further optimization and refinement of our system could enhance its accuracy and usability, ultimately providing users with a seamless and enjoyable movie discovery experience.

Discussion

Interpreting the result

Our movie recommendation system showcases the effective integration of clustering techniques and natural language processing to deliver personalized movie suggestions based on user input. Through the utilization of KMeans clustering along with dimensionality reduction methods like PCA and LDA, we successfully group similar movies, enabling a streamlined recommendation process. By incorporating user preferences via simple one-word prompts, our system significantly reduces the time and effort users typically expend on selecting movies, thereby enhancing their overall viewing experience. This approach demonstrates the system's ability to adapt to user preferences and provide relevant recommendations, ultimately addressing the challenges associated with content discovery in the digital streaming era.

Limitations

Some limitations that we faced when trying to implement our project was that if we decided to add another variable into the recommendation function as to look at the genre as well as the synopsis at the same time when trying to recommend a movie, it appeared that inputting a genre would overshadow the recommendation of the synopsis. That is to say if someone put ‘comedy’ and ‘basketball’ at the same time, more generic comedy movies would be recommended over comedic basketball movies, so due to a lack of time we could not implement a way to have both genre and synopsis have equal weight.

Ethics & Privacy

In developing a movie recommendation system, it's crucial to navigate the potential ethical and privacy concerns that accompany the collection, analysis, and application of user data. In our project to develop a movie recommendation system, we utilize a dataset sourced from Kaggle, an open-source platform known for providing a wealth of datasets for various machine learning projects. This particular dataset comprises detailed information about movies, including genres, ratings, crew details, and overviews, but notably does not contain any direct user information. The open-source nature of the data and the absence of personal user data alleviate privacy concerns. Some ethical concerns that we may have is the impact of our recommendation algorithm on future movies, if this project ends up being very successful and popular in recommending movies, movie companies may use this as a basis to try to make the most profitable movie, instead of a movie that they put their heart and soul into that ended up being very popular. This would end up being a detriment to society as the quality of media would be lowered and become something that panders to the most profitable group instead of being a medium to express artistic vision.

Conclusion

Our results display the potential of our movie recommendation system to revolutionize the way users discover and select movies, offering personalized suggestions based on their interests. By effectively clustering movie synopsis and incorporating user input, our system streamlines the content discovery process, mitigating the challenges of choice overload prevalent in the streaming era. Furthermore, our exploration of alternative techniques highlights the iterative nature of model development, emphasizing the need for ongoing refinement and adaptation to ensure optimal performance. In the broader context of recommender systems, our work contributes to the evolving landscape of content recommendation, aligning with the industry's demand for innovative solutions that enhance user experiences in digital content platforms. Looking ahead, future work could focus on fine-tuning algorithms, integrating user feedback mechanisms, and expanding the system's capabilities to cater to diverse user preferences and viewing habits.

Footnotes

1.^: Marcy.franklin. (2019, November 20). *How much time do you spend picking out what to watch next?*. Vox. https://www.vox.com/ad/20974139/streaming-content-movies-tv-shows-algorithm-human-choice
2.^: Consumerlab Report on TV and Media 2017 - Ericsson. (n.d.). https://www.ericsson.com/en/reports-and-papers/consumerlab/reports/tv-and-media-2017
3.^:$71K-$197K recommender systems jobs (now hiring) ... (n.d.-a). https://www.ziprecruiter.com/Jobs/Recommender-Systems