Intro to Data Science Final Project

Movie Recommendation System

Amodhiniram Kanagarajan

Github Link

Introduction:

What question am I trying to answer?
With digital content expanding at explosive levels, end users are faced with daunting choices when trying to choose a movie to watch. The purpose of this project is to see if we can build a good system that recommends movies a user will enjoy, either by the content of movies they've previously watched or the preferences of individuals like the user. This is an important question to many real-world use cases like Netflix, Hulu, and Amazon Prime, and ties well into the growing need for personalization.

How do I plan to answer it?
To answer this, I developed two types of movie recommendation systems using unsupervised learning techniques. First, a content-filtering approach inspects the attributes of each movie like genres, keywords, cast, and director and suggests similar movies based on cosine similarity of TF-IDF vectorized text data.
Second, a collaborative filtering model uses a user-movie rating matrix and a K-Nearest Neighbors (KNN) algorithm to suggest movies based on similar user behavior. The combination of these two systems offers insight into both content-based and behavior-based recommendation patterns.

How does this approach relate to the lectures/papers discussed?
I have studied key data science concepts like feature engineering, similarity measures, text vectorization (TF-IDF), and some unsupervised learning methods like clustering and KNN during the course.
The project consolidates these concepts into a tangible implementation. Specifically, the content-based model utilizes TF-IDF, something that I studied during the NLP module, and the collaborative filtering approach utilizes KNN, an unsupervised technique that I studied in a very detailed manner through our machine learning lectures.
The handling of large tabular data, feature extraction from unstructured JSON columns, and sparse matrix manipulation all represent techniques that I have studied throughout this semester. This two-method recommendation project illustrates the application of course material to an actual problem, technical competence as well as greater knowledge of model selection, evaluation, and trade-offs in unsupervised learning.

Motivation:

Why is my project important? Why am I excited about it?
Recommendation systems are the core of the way users engage with material on the internet. Seeing a movie, listening to music, or web shopping. Targeted recommendations enhance the experience and drive engagement. My project is important because it mimics the nature of such systems, providing insight into how Netflix and Amazon Prime type of websites personalize user experience. From a student's perspective, it's not only fun to apply ideas about unsupervised learning from theoretical abstraction to a tangible, real-world instance that can make intelligent, coherent decisions based on data, but also to work on systems that help people make better choices.

What are some existing questions in the area?
There are some open questions in the area of recommender systems:

- How do we personalize recommendations for new products or users with limited data?
- How do we balance popularity bias to suggest less-rated but potentially interesting movies as well?

- How do we effectively combine collaborative and content-based approaches into a hybrid model?
- How can we evaluate the usefulness or diversity of recommendations without explicit user feedback?

Are there any prior related works? Provide a brief summary.
GroupLens Research's MovieLens project at the University of Minnesota is one of the first datasets in recommender system research and has inspired much collaborative filtering and matrix factorization research. A number of the other classic works include the Netflix Prize competition models that tried ensemble methods and deep learning collaborative filtering. All of these, such as content-based filtering using TF-IDF, latent semantic analysis (LSA), and collaborative filtering using user-user or item-item KNN, have been extensively studied in research literature and are the theoretical backgrounds upon which models in this project are constructed. I borrowed inspiration from such systems.

Method:

What dataset did I use?
I utilized "The Movies Dataset" from Kaggle, which consists of some related CSV files: movies_metadata.csv, ratings_small.csv, credits.csv, keywords.csv, and links_small.csv. These data sets provide rich metadata for thousands of films and more than 100,000 user ratings.

	adult	belongs_to_collection	budget	genres	homepage
1	False	lEDUrTdcaafOMKUq.jpg'}	30000000	: 10751, 'name': 'Family'}]	story.disney.com/toy-story
2	False		65000000	: 10751, 'name': 'Family'}]	
3	False	oFS7qwsQHW1uV8u.jpg'}	0	id': 35, 'name': 'Comedy'}]	
4	False		16000000	0749, 'name': 'Romance'}]	
5	False	lChBpLEbJEmzUydk.jpg'}	0	id': 35, 'name': 'Comedy'}]	
6	False		60000000	{'id': 53, 'name': 'Thriller'}]	
7	False		58000000	0749, 'name': 'Romance'}]	
8	False		0	: 10751, 'name': 'Family'}]	
9	False		35000000	{'id': 53, 'name': 'Thriller'}]	
10	False	/dXOZfJPdarlUGOsk.jpg'}	58000000	{'id': 53, 'name': 'Thriller'}]	ew/movie/757/Goldeneye/
11	False		62000000	0749, 'name': 'Romance'}]	
12	False		0	{'id': 27, 'name': 'Horror'}]	
13	False	o1st1KyHA3cVnO2G.jpg'}	0	': 12, 'name': 'Adventure'}]	
14	False		44000000	{'id': 18, 'name': 'Drama'}]	
15	False		98000000	': 12, 'name': 'Adventure'}]	
16	False		52000000	, {'id': 80, 'name': 'Crime'}]	
17	False		16500000	0749, 'name': 'Romance'}]	

Figure 1: movies_metadata.csv

	userId	movieId	rating	timestamp
1	1	31	2.5	1260759144
2	1	1029	3.0	1260759179
3	1	1061	3.0	1260759182
4	1	1129	2.0	1260759185
5	1	1172	4.0	1260759205
6	1	1263	2.0	1260759151
7	1	1287	2.0	1260759187
8	1	1293	2.0	1260759148
9	1	1339	3.5	1260759125
10	1	1343	2.0	1260759131
11	1	1371	2.5	1260759135
12	1	1405	1.0	1260759203
13	1	1953	4.0	1260759191
14	1	2105	4.0	1260759139

Figure 2: ratings_small.csv

	id	keywords
1	862	name': 'toy comes to life'}]
2	8844	91, 'name': 'giant insect'}]
3	15602	08510, 'name': 'old men'}]
4	31357	3455, 'name': 'chick flick'}]
5	11862	21, 'name': 'gynecologist'}]
6	949	983, 'name': 'crime epic'}]
7	11860	626, 'name': 'millionaire'}]
8	45325	[]
9	9091	, 'name': 'vice president'}]
10	710	3008, 'name': 'red army'}]
11	9087	e': 'wildlife conservation'}]
12	12110	d': 11931, 'name': 'spoof'}]
13	21032	371, 'name': 'frozen lake'}]
14	10858	'name': 'historical figure'}]
15	1408	d': 12988, 'name': 'pirate'}]
16	524	ame': 'illegal prostitution'}]
17	4584	66512, 'name': 'decorum'}]

Figure 3: keywords.csv

What form does this data have? Is it images, raw text, tabular, etc? What are the features?
The data is tabular, consisting of a mix of structured fields (e.g., movieId, userId, and rating) and unstructured fields (e.g., overview, genres, keywords, cast, and crew). The majority of columns use stringified JSON format to hold nested data, requiring custom parsing. The features span several data types:

Textual: overview, keywords, genres
Categorical: cast, crew, userId, movieId
Numerical: rating, popularity, vote_average, runtime

What analysis did I do?
This project involved two main unsupervised learning pipelines:

Content-Based Filtering:
I first **preprocessed and cleaned the metadata**. I **extracted JSON-formatted fields** to get useful data such as the top 3 cast members, director, genres, and keywords. I then created a "soup" - a single text feature that combines all these fields in a bag-of-words format. With **TF-IDF vectorization**, I transformed the soup into a numerical vector per movie. Then, **cosine similarity** was calculated for all movie vectors to determine most similar movies regarding content. I developed a recommendation function that takes a movie title as input and returns the most similar 10 movies according to this content profile.

Collaborative Filtering:
I preprocessed ratings data to construct a user-item matrix where the rows represent users and the columns represent movie IDs. Due to the inherent sparsity of the data (most users rate very few movies), I pruned the dataset in such a way that I have only those movies which have been rated by 50 or more users. This removed extraneous noise and left sufficient overlap within the rating matrix. I executed **K-Nearest Neighbors (KNN)** with cosine distance on this matrix (transposed to movies as rows) to obtain item-based neighbors. I defined a function that takes a movie title and returns similar movies based on other users' behavior when rating.
While developing this model, I initially encountered a MemoryError while importing the whole credits.csv and keywords.csv files. These files contained over 40,000 rows and contained big stringified JSON fields, which were greater than the local memory when I was performing merges and transformations. To resolve this, I only loaded the correct columns (id, cast, crew, keywords) and utilized the n-rows parameter in pandas to reduce the dataset to the first 5000 rows. This trade-off still kept a large percentage of movies but made the model runnable.

Additionally, I implemented fuzzy title matching for bettering the performance of the collaborative filtering model. If the user gives a partially or misnamed movie, the function will return the best match available in the matrix and provide recommendations accordingly.

This analogy allowed me to delve into commonalities and distinctions between two classes of unsupervised learning algorithms—one based on inherent item characteristics, and one based on shared user behavior patterns. It also gave me insight into trade-offs related to explainability, scalability, and dependency on user data.

Results:

What results did your analysis show? Visualize them if possible.

Our analysis provided some clear, data-driven findings supported by numerical trends and visualizations:

Finding 1: Blockbuster Movies Overwhelm Ratings and Control Collaborative Filtering

I noticed that some movies had the majority of ratings. Forrest Gump, The Shawshank Redemption, and Pulp Fiction were some of the top-rated ones, and therefore most likely to be recommended under collaborative filtering, purely due to data density.

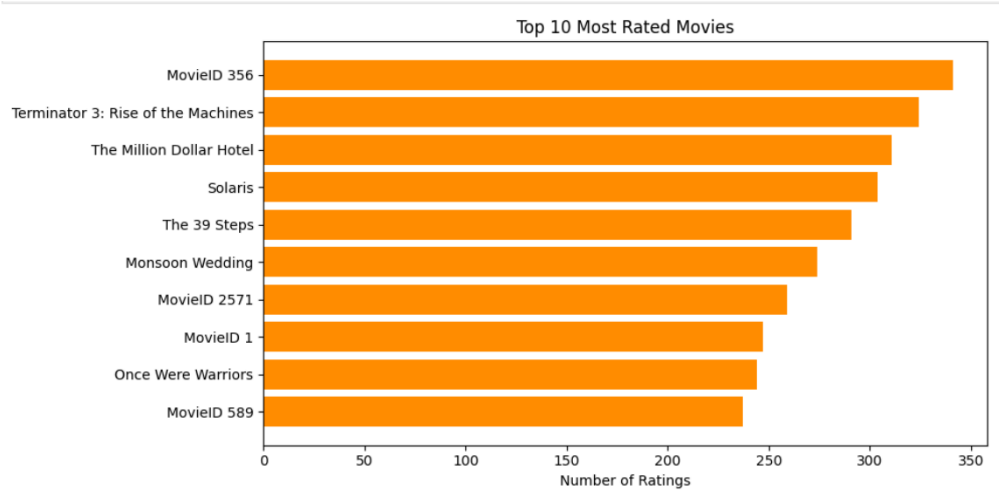

Figure 3: Top 10 Most Rated Movies

The bar chart illustrates the movies with the highest number of user ratings, clearly highlighting a few titles that stand out as dominant in the dataset.

Finding 2: Genre Distribution is Skewed Towards Drama and Comedy

From the content-based point of view, I discovered that the majority of the movies in the dataset fall into a limited number of genres. That is, Drama and Comedy appeared much more frequently than other genres.

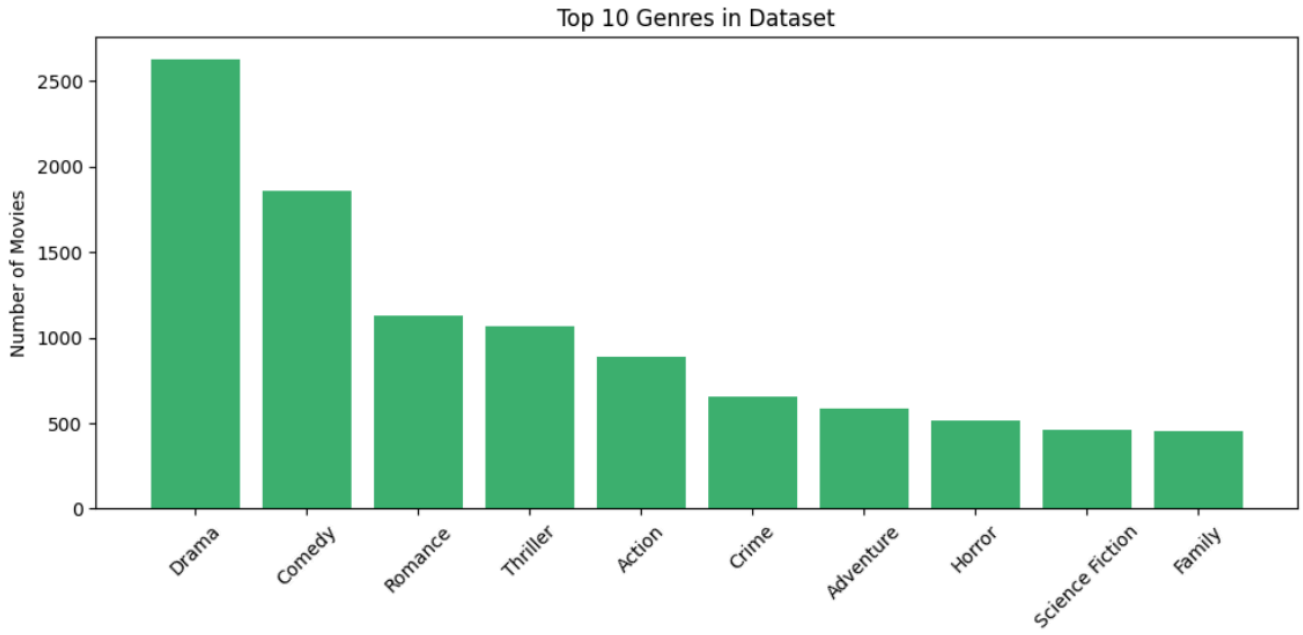

Figure 4: Top 10 Genres in Dataset

This is why most content-based recommendations lean heavily towards these genres, regardless of input movie.

Finding 3: Thematic Similarity Reflected in Content-Based Results

This content-based recommender performed best in making thematic and stylistic matches. For instance, a request for Toy Story gave back movies such as Small Soldiers, Toys, and Child's play - all animated, family films that feature keyword and visual storyline similarity.

```
# Try it
print(get_recommendations('Toy Story'))
```

```
['Toy Story 2', 'Small Soldiers', 'Toys', "Child's Play 3", "Child's Play", 'Stardust Memories', 'Take the Money and Run', 'Manhattan',
'Shadows and Fog', 'Deconstructing Harry']
```

Figure 5: Code snippet

These results validate that the TF-IDF + cosine similarity model was indeed identifying content-based patterns.

Finding 4: Collaborative Filtering Captures Behavior Patterns, Not Content

In contrast, introducing Batman Returns to the collaborative framework returned Reservoir Dogs, The Hours, and To Kill a Mockingbird. Though their themes were unrelated, they were both box office hits co-rated by fans of Batman Returns, thereby illustrating the behavior-driven nature of collaborative filtering.

```
print(recommend_by_title("batman"))
```

```
Did you mean: 'Batman Returns'?

['Silent Hill', 'To Kill a Mockingbird', 'Reservoir Dogs', 'Monsoon Wedding', 'Wag the Dog', 'Bang, Boom, Bang', 'The Hours', 'A Nightm
are on Elm Street']
```

Figure 6: Code snippet

This confirmed that the understanding that collaborative filtering does not take into account movie content - it mimics user behavior.

Finding 5: Keywords Reveal Thematic Interconnections in Movie Plots
In order to look at more significant patterns in the way themes are matched between films, I created a keyword co-occurrence heatmap. Unlike simply showing frequent words, this graph displays how often specific keywords appear together within the same films.

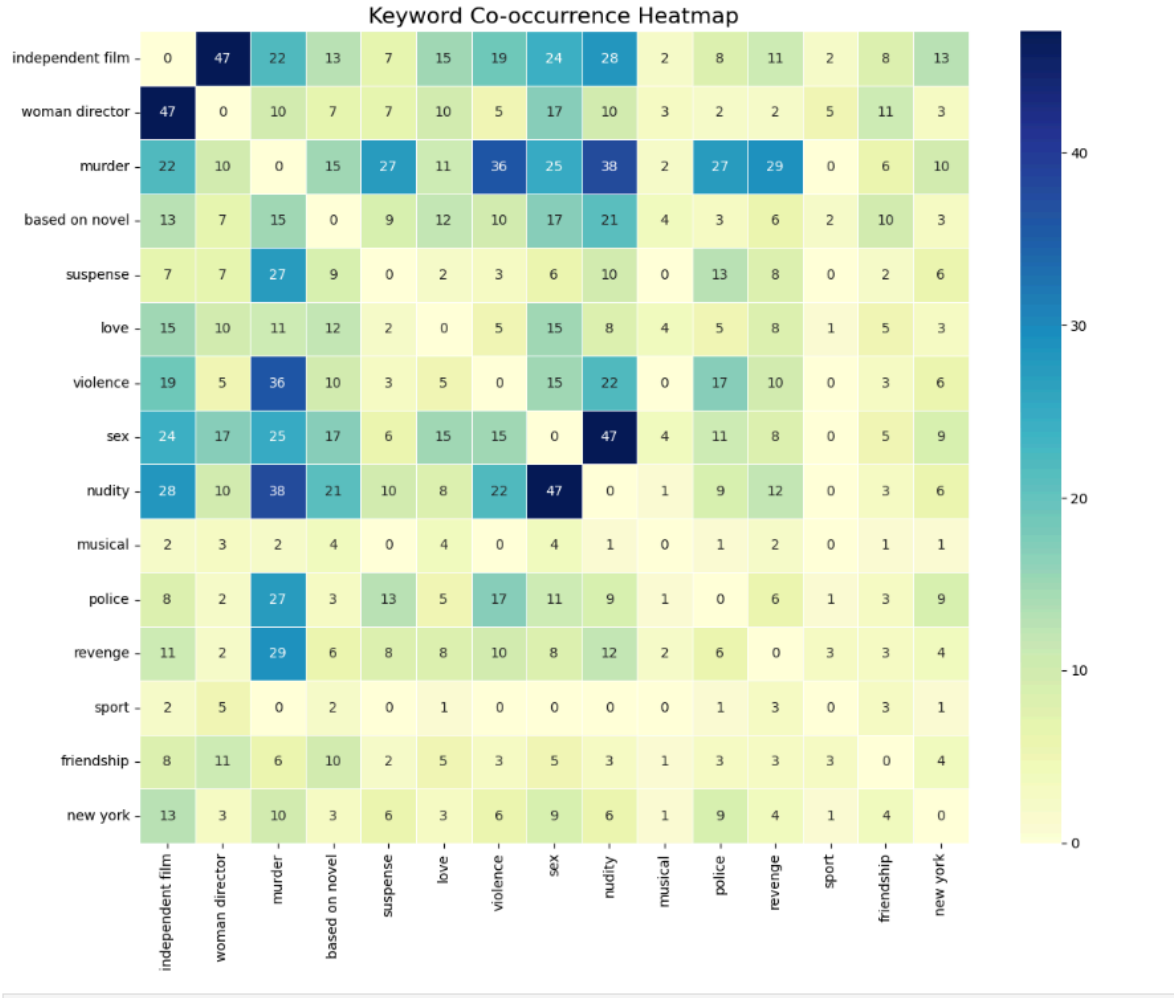

Figure 7: Keyword Co-occurrence Heatmap

The heatmap reveals rich narrative clusters:

- "Murder", "revenge", and "violence" are highly correlated, suggesting many films revolve around justice, crime, or retribution themes.
- "Independent film" and "woman director" co-occur frequently, which implies the presence of frequent patterns in tagging or thematic focus in independent films.
- Emotional themes like "love", "friendship", and "suspense" co-occur less frequently, which may imply broader distribution across diverse genres and not co-clustering around specific themes.
- Certain words like "sport" and "musical" co-occur more in isolation, which might imply specialized themes.

These observations shed light on how content-based suggestions can happen to group similar films by their same emotional theme or narrative design. It also speaks to why relationships between keywords must be taken into consideration rather than as solo labels.

What new questions do these results raise, and how can they be addressed by further analysis?

- Would a hybrid model that combines both content and behavior perform better than either model in isolation?
- How do I reduce popularity bias and increase diversity in suggestions?
- Can I develop an explainable AI system that tells us why a specific movie is being recommended?

These results demonstrate that both models are great and not flawless, and their combination may provide us with the best of both worlds.

Discussion:

What outcome did I expect from my results?
I was expecting that the content-based model would recommend movies with high thematic and stylistic similarity—particularly in terms of genres, keywords, and cast similarity. For the collaborative model, I was hoping that the system would produce user-preference-based recommendations with potential influence by blockbuster movies.

How did your actual results differ from your expected results?
In most cases, the results were as expected. Content-based filtering correctly retrieved films with similar content, and collaborative filtering retrieved films liked by users with similar tastes. However, the collaborative model more extensively exhibited popularity bias than anticipated, with the same top-rated films appearing time and time again.
Moreover, in some boundary cases, the collaborative model recommended films that were not thematically related, further supporting its behavior-driven rather than content-based justification.

If my final report differs from my proposed project, discuss the differences, why I made certain changes, and the bottlenecks that prevented me from proceeding with the proposed project?

The initial project I proposed was to build a book recommendation system in partnership. Since there were coordination changes in the group, I decided to work individually and changed to build a movie recommendation system. This change of dataset and project direction allowed me to continue with unsupervised learning approaches while maintaining the underlying learning objectives of the original proposal.

On the technical bottleneck end, I encountered memory problems when attempting to load full credits.csv and keywords.csv files. They contained high-dimensional, complex data (stringified JSON) and required many computational resources. As a compromise, I reduced the scope of the dataset by limiting rows and taking only required columns into account. This allowed me to focus on fully understanding the models and ensuring quality recommendations over user interface design.