Integrated Entertainment: Improving Video Streaming

A Technical Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Cameron Woodward

with

Bendert Stansell

May 7, 2021

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Cameron Woodward

Technical advisor: Miaomiao Zhang, Department of Computer Science

Integrated Entertainment: Improving Video Streaming

Cameron Woodward
Computer Science
University of Virginia
Charlottesville, VA
cnw2bx@virginia.edu

Bendert Stansell
Computer Science
University of Virginia
Charlottesville, VA
bjs4sy@virginia.edu

ABSTRACT

With the continued proliferation of streaming services, watching TV has become scattered across many streaming services competing for your attention. When deciding what to watch you sometimes find yourself wandering through multiple different apps for different streaming platforms, and it can be easy to forget where the content you are looking for lives. In the days of cable TV, there were helpful guides where you could see what was playing on each channel in one place by simply scrolling through. A centralized place to find all TV shows from different streaming services exists on some platforms (Amazon Firestick, Apple TV, Google TV), but it is lacking on Xbox One. With this project we will connect the streaming services so that Xbox users can have a centralized location to find (and even have suggested to them) what they're looking for. Our app takes a title and queries each platform to see where the show is hosted and returns the results. As well as connecting these platforms, our app uses Machine Learning techniques to customize suggested shows for the user. This app will improve not only the efficiency of searching for shows, but the quality of service provided to the user. By reducing the time for Xbox users to find shows, it will give them more time to watch or do whatever they please. This directly impacts the streaming market because this enhanced and easy to use experience will cause current viewers to watch more, while simultaneously attracting additional viewers.

INTRODUCTION

With streaming subscriptions surpassing cable packages, when deciding what to watch viewers often find themselves wandering through multiple different apps for different streaming platforms [3]. There are a couple slightly different reasons that users meander across streaming services. It could be that all of your friends have been watching the show "Friends." You want to watch too. So you go to Netflix, but unbeknownst to you, "Friends" recently had their contract with Netflix expire and has moved to HBO Max. Another scenario is that you sit down on a cold winter night wanting to watch a great Sci-Fi Action movie that you've never seen before. You first look through Amazon Prime, and you find a few that you might want to watch but aren't sure. You then proceed to look through 4 other streaming services looking for the perfect movie. In both of these common scenarios, having one place to find all the streaming content streamlines the process and saves the user precious time. The improved convenience is obvious to anyone who uses multiple streaming services on a regular basis, and the benefits are even more dramatic on an Xbox because the typing interface, voice control, or other typing input are lacking. Typing input is slow because most users manipulate an analog stick to find each letter, which can compound to minutes or even hours over many iterations. (keyboard support was added by Microsoft in late 2019, but each Xbox app is responsible for adopting support). Typing in a search only one time will save not only the amount of time to close and open multiple apps, but also the non-trivial amount of time it takes to type out a search.

RELATED WORK

Right now, the market has products such as the Amazon Firestick and Apple TV which provide a similar service as our app. They provide a central "hub" of different apps, but they lack integration across platforms. It seems likely that these services will never migrate over to any other platform. This is because they are used as a headline feature for people to buy the product that has them. For example, you must buy an Amazon Firestick, Apple TV, or chromecast (Google TV) to access any of these incredibly useful services. There is a similar service on some android smart TVs, but it is often not as robust as these previously mentioned services. For example, LG android TVs have a universal search feature across streaming services, but the UI is not great. Also, this LG service only provides the searching feature and does not

have a central place that displays content. In other words, this provides the universal search feature that we are after, but does not provide the universal recommendation feature (as well as not being on the same platform). Additionally, there are some TVs that have one of these services "built-in." However, this feature is only on select TVs and is not an app that can be downloaded. Admittedly, Google TV launched only a few months ago, so the way they choose to develop this product is unclear. It could be that they try to have Google TV available everywhere in the future. There is another service that is similar called Plex. Plex does have an application for almost every platform, including Xbox, but it is slightly different. Plex allows users to stream a movie from any device to another device, and is essentially a home media server. For example, Plex allows the user to buy the new Star Wars movie on google play, download it on their laptop, then stream it to their smartphone or console at any time. So while Plex does allow a streaming hub on Xbox, it does not fulfill the same niche as our Xbox app will.

1 **SYSTEM DESIGN**

The successful implementation of this project will incorporate many computer science methods and techniques. To contact individual stream providers (such as Netflix, Amazon, etc.), we will use an Application Programming Interface (API). This will allow us to query each provider and see if they host the specific content related to a search. In order to recommend shows for users we will aggregate a lot of data and develop a machine learning model to group movies. Preliminary data will be sourced from Kaggle, which contains a dataset of 45,000+ movies and TV shows with title, cast, ratings, genre, and keywords. This will suffice for general predictions, but once the customer has used the app, we will apply their specific data to personalize recommendations further. As for the model, we will use a clustering algorithm. Specifically, we will use K-means clustering, which will provide us with groups of movies that are similar to each other.

1.1 **Building the model**

To create the K-means model, we first downloaded and imported the dataset from Kaggle into Google Colab. Then we took a deeper look into the data to understand the problem more and see any patterns or trends we could see. The dataset had 20 features with which we could train our model on. Some of these features were irrelevant, therefore we picked 9 features that were most useful for grouping movies together. The nine categories of features included were budget, genres, keywords, popularity, revenue, runtime, title, vote average, and vote count.

- budget: the money for the movie in dollars
- genres: a list of dictionaries of all the associated genres
- keywords: a dictionary of the most common strings in the movie and their frequency
- popularity: a score assigned by TMDB
- revenue: total money made from the movie in dollars
- runtime: how long the movie is in minutes
- title: the name of the movie
- vote average: average rating of the movie from TMDB users
- vote count: number of TMDB users that voted

Once we had these categories, we plotted histograms of each feature to see its distribution. Most were approximately normally distributed, but popularity, budget, and revenue were skewed to the right due to outliers like Marvel and other extremely big budget movies. After we saw the individual data, we moved on to examining the correlation between each feature. We started by creating a correlation matrix and found that vote average, vote count, and popularity all had a strong positive correlation. To visualize these correlations, we graphed scatter plots of all the features in combination with every other feature. These graphs showed us that revenue and budget were also correlated positively. Our next step was to calculate general statistics of the data. These are summarized below:

```
[ ] movies.describe()
```

	budget	popularity	revenue	runtime	vote_average	vote_count
count	4.803000e+03	4803.000000	4.803000e+03	4801.000000	4803.000000	4803.000000
mean	2.904504e+07	21.492301	8.226064e+07	106.875859	6.092172	690.217989
std	4.072239e+07	31.816650	1.628571e+08	22.611935	1.194612	1234.585891
min	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000	0.000000
25%	7.900000e+05	4.668070	0.000000e+00	94.000000	5.600000	54.000000
50%	1.500000e+07	12.921594	1.917000e+07	103.000000	6.200000	235.000000
75%	4.000000e+07	28.313505	9.291719e+07	118.000000	6.800000	737.000000
max	3.800000e+08	875.581305	2.787965e+09	338.000000	10.000000	13752.000000

Figure 1: **basic statistics of the movie's dataset**

Our last step of preprocessing was to clean our data and prepare it for training our model. We first split our data into training, testing, and validations sets. In order to start cleaning the data we looked for any rows that were

null. There ended up being a few rows where the runtime was null. In an effort to deal with this and make our data uniform, we created a pipeline. We added layers to this pipeline to deal with numerical and categorical data. For the numerical pipeline we used an imputer to fix the issue of null runtimes as well as a standard scaler to normalize our data to produce better results. The categorical pipeline called the numerical one and then used OneHotEncoder to transform our categorical data to numbers so that we can operate on it. Lastly, we broke up the training and testing sets into categorical and numerical and ran them through the pipeline to produce clean and prepared data.

After we successfully cleaned our data it was time to build our model. Using Sci-kit learn KMeans implementation we initialized a model with an arbitrary number of clusters to see what was returned and what we could do with it. We found out that after running fit and predict on our model with the cleaned dataset we got the centroids of each cluster back. We decided to plot those to see where each cluster is. After this we decided to employ a strategy of "finding the elbow" to optimize the number of clusters to use. We did this by training models with cluster size equal to 4 all the way up to 50. We then plotted the inertia from each of the models to find the one that caused the sharpest change in inertia. This ended up being a cluster size of 17 which was seen from the elbow and supported by silhouette scores as see below:

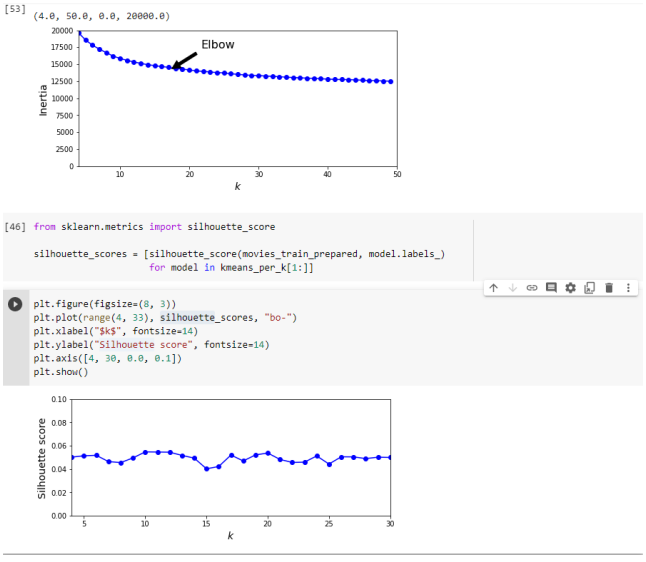

Figure 2: **plots of the inertia and the silhouette scores on the top and bottom, respectively**

We then trained a separate model with the optimum number of clusters and plotted it's centroids as seen below:

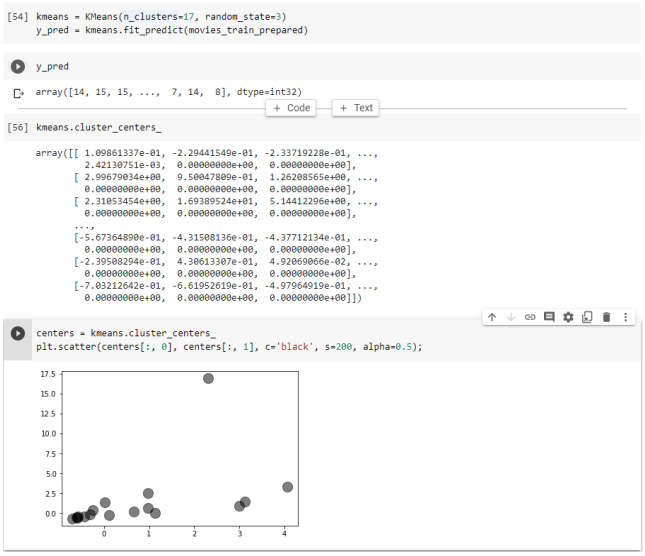

Figure 3: **centroids output from the optimized model**

1.2 Using API's

In order to communicate with the stream providers and see if the desired content is hosted on their servers, we used RapidAPI. This allowed us to query Hulu, Amazon, and Netflix which don't have publicly available API's. We first created a free account on RapidAPI. Then we found the closest endpoint to communicate with, which was */api.cgi* for Netflix. From there we utilized the *requests* library in Python to create and send GET requests to this endpoint. We successfully received response codes of 200 along with a JSON response. This JSON response included the title, its NetflixID, ratings, runtime, and type. We also tried to request titles that weren't hosted and got back 404 responses. This allowed us to create a python script that takes a search from a user and creates a GET header, sends the request to our connected endpoint, and then returns whether it was found or not.

1.3 Xbox App

For the frontend of the Xbox app, we are able to use the Xbox One development mode. The interface of the app mimics that of any modern streaming service, where the majority of the screen is filled with rows of movies and

TV shows sorted in categories that can be scrolled through, such as "Sci-Fi" or "Popular in the US." Additionally, there is a search bar at the top to look up a specific title. When a user selects the content that they want to play, the app will redirect them to the player from the necessary app. Of course, content from a specific provider will only be included if the user is logged in to that provider on this device.

PROCEDURE

The app is used almost exactly the same as any singular streaming app currently available on Xbox. To use our app, users must download it from the Microsoft store onto their Xbox. From there they can create an account, which will help with the recommendations. When in the app, the user can input a title of a show or movie they want to watch and our app handles the rest. The result of the query is displayed on the screen. It is possible that the exact title will not exist on any streaming platform, in which case there will be similar titles that will be displayed. There will also be a browse section where the user can simply scroll through the recommended titles until they find one that they would like. This will be broken into sections such as genres, recently watched, titles similar to liked titles, and most popular. Once a title is selected, the user will then be seamlessly redirected to the player from the streaming service providing the desired content. For any particular streaming service to be included, the user will have to be logged into their account on this device.

Other stakeholders of our app are the stream providers themselves. They may not directly interact with our users, but our app does affect them. If our app was used by many people then that will imply more traffic and users to the providers. This may lead to future partnerships where we work directly with the providers to update content and data to improve both apps. They will still get the data generated from the users interacting with their content because our app will be interacting with the content using APIs.

RESULTS

This project aims to enhance the individual streaming experience by increasing efficiency and personalization. Having a central location will save users the valuable time it takes to type and search for a title across many different platforms. When searching for a specific show, the user saves about 5 to 10 minutes per show. When browsing, the time saved varies widely per user, ranging from about 5 to 30 minutes saved each time. Using machine learning to personalize the content will encourage the user to watch programs they enjoy which they may not have found or searched for independently. A centralized platform coupled with customized recommendations will achieve the goal of enhancing the overall experience by making it easier, efficient, and tailored.

The end product will be an Xbox app that streamlines video searching, providing the user with an easier and customized experience. This will be a relief to users, but the stream providers will also benefit. Increased customer satisfaction added with extra time to watch shows instead of searching will drive subscription and retention rates up. Better quality of service and increased subscription rates may open the door for potential partnerships with the providers.

CONCLUSION

We designed an Xbox One app that simplifies and improves the video streaming experience. Instead of rotating through multiple separate apps searching for content, users can find anything they would like in this one place. They will save time when searching for a specific title or when browsing to see what new show they would like to watch. This is a central hub to find all streaming content from any services that the user enjoys. Content will be recommended to the user based on items that they have previously enjoyed. This content is recommended based on a K means algorithm that filters all of the content into separate groups. Once a particular show is selected, the app will open up the player from the appropriate provider (if you clicked on The Mandalorian, the Disney+ player would open). The UI is not revolutionary, the machine learning algorithm is not revolutionary, but the convenience is revolutionary.

FUTURE WORKS

There is no reason that this product should be limited to the Xbox One. Ideally this app could be expanded for use on Windows, MacOS, Android (with special integration with android TVs), iOS, and PlayStation. Virtually everyone could benefit from this service because of the universality of streaming. It is not hard to imagine a

market for some sort of similar application on each of these markets. A successful implementation of this project would also likely require special access to each of these streaming services, which is difficult to obtain (note that the parties who have been able to do this are Amazon, Google, and Apple). Given more time, it is possible that the companies would buy into the idea of this hub for streaming services. On top of expanding into other platforms, with extra time additional features could be added. One potential addition could be separate profiles on a single device, which is common with streaming services. Another feature could be censorship of certain material as well as kid accounts. A benefit that comes with more time is an improved machine learning model, the longer it runs and the more data it receives will constantly improve it. Deploying a strategy similar to Netflix, we could hold public competitions each year to improve upon or create a better model for recommendations [4]. This would not only inspire innovation and competition, but also provide a great way to notice fellow computer scientists for their hard work.

ACKNOWLEDGMENTS

We would like to give a big thank you to Professor Miaomiao and Professor Worthy of the Computer Science department at UVA for helping us along the way. They helped us find a direction for this project from the beginning and gave us advice and ideas when we were unsure of what to do next. Their creative solutions coupled with their extensive experience gave us a much-needed perspective.

REFERENCES

[1] https://tv.google/
[2] https://www.apple.com/tv/
[3] Miller, J. (2019, March 23). Streaming subscriptions surpassed cable worldwide in 2018. https://www.techspot.com/news/79333-streaming-subscriptions-surpassed-cable-worldwide-2018.html
[4] Chen, E. (2011). Winning the Netflix Prize: A Summary. https://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/