

VYTAUTAS MAGNUS UNIVERSITY
FACULTY OF INFORMATICS
DEPARTMENT OF APPLIED INFORMATICS

Satima Valsony

CONTENT-BASED K-DRAMAS RECOMMENDATION WEBSITE

Bachelor Thesis

Informatics Systems study program, state code 6121BX016
Study field Informatics

Supervisor Aušra Vidugirienė

degree, name, surname — signature, date

Defended prof. dr. Tomas Krilavičius

Dean of Faculty — signature, date

Kaunas, 2023

CONTENT

LIST OF ABBREVIATIONS 4
ABSTRACT 5
SANTRAUKA 6
1 INTRODUCTION 7
2 ANALYSIS OF LITERATURE, TOPIC, AND METHODS 10
2.1 Literature analysis 10
2.2 Topic analysis 12
2.3 Methods analysis 14
2.3.1 Web scrapping 14
2.3.2 Content-based filtering 16
2.3.3 TF-IDF 16
2.3.4 Similarity measuring 17
2.3.5 Adamic Adar measure 18
2.3.6 Graph representation of network 18
3 RECOMMENDATION ENGINE MODELLING 20
3.1 Architecture 20
3.2 Data gathering and web scrapping 21
3.3 Data preprocessing 24
3.4 Exploratory data analysis 25
3.5 Model building 26
3.6 Web application building 30
3.7 Web application improvement with integrated dynamic graph 32
3.8 Testing 33
4 RESULTS AND CONCLUSIONS 35
4.1 Results 35
4.2 Conclusions 36

4.3 Perspectives 37
5 References 38
6 APPENDICES 40
6.1 Sources 40

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CBF	Content Based Filtering
CSV	Comma Separated Values
EDA	Exploratory Data Analysis
HTTP	Hyper Transfer Protocol
HTML	Hypertext Markup language
k-drama	Korean drama series
KMDb	Korean Movies Database
LDA	Latent Dirichlet allocation
ML	Machine Learning
NLP	Natural Language Processing
TF-IDF	Term Frequency Inverse Document Frequency
XML	Extensible Markup Language

ABSTRACT

Author	Satima Valsony
Title	Content-based K-dramas Recommendation Website
Supervisor	Aušra Vidugirienė
Presented at	Vytautas Magnus University, Faculty of Informatics, Kaunas (08.06.2023)
Number of pages	40
Number of appendices	2

Because of the huge amount of information and diversity of options, modern people are facing a new problem in this century - an overwhelming number of choices. Even while choosing a next movie to watch, which seemed to be an easy question, we frequently feel uncertainty and confusion which can lead to decision fatigue and frustration. Fortunately thanks to the growth of AI tools, nowadays there are various types of movie recommendation engines based on different algorithms and filtering techniques. But these endless varieties of recommendation systems are built by using several well-known models, for the majority of the audience and trained by a couple of existing large open-source datasets such as TMDB Movie, Netflix dataset or Movie lens which are already accessible in Kaggle or other platforms.

This bachelor thesis covers the idea of building a content-based Korean drama recommendation system with our own collected dataset and developed by a new approach to model building, and finally with the state-of-the-art Python library for web app development.

Korean dramas are becoming more and more popular nowadays, but it's impossible to find any AI-powered recommendation system for this specific genre of television series with a unique structure and features. The dataset of Korean dramas was scrapped from different sources, and after it was cleaned and preprocessed, the data was represented and saved as an indirected graph where each Korean drama is stored as a node in space. Each node in space is connected by an edge with another node representing the relationship between similar Korean dramas. In order to measure the distance between nodes Adamic Adar measure was used. As higher the measure, as closer the two nodes or dramas.

SANTRAUKA

Autorius	Satima Valsony
Pavadinimas	Turiniu pagrįstas K-dramų rekomendavimo tinklapis
Vadovas	Aušra Vidugirienė
Darbas pristatytas	Vytauto Didžiojo Universitetas, Informatikos fakultetas, Kaunas (08.06.2023)
Puslapių skaičius	40
Priedų skaičius	2

Dėl didžiulio kiekio informacijos ir pasirinkimo įvairovės, šiuolaikiniai žmonės susiduria su nauja problema - dideliu pasirinkimų skaičiumi. Net rinkdamiesi kokį kitą filmą žiūrėti, kas atrodo, neturėtų sukelti daug sunkumų, dažnai jaučiamės sutrikę, kas gali privesti prie nuovargio ar nusivylimo. Laimei, AI įrankių tobulėjimo dėka, dabar yra įvairių tipų kino filmams skirtų rekomendavimo sistemų, pagrįstų skirtingais filtravimo algoritmais . Šios rekomendavimo sistemos yra sukurtos, naudojant kelis žinomus modelius, apmokytus su keliomis atviros prieigos duomenų bazėmis kaip TMDB Movie, Netflix ar Movie, kas jau yra prieinama Kaggle ar kitose platformose.

Šiame bakalauriniame darbe kuriama turiniu pagrįsta Korėjos dramų rekomendavimo sistema su mano pačios surinktu duomenų rinkiniu, naudojant naujo tipo modelį, bei pasitelkiant dabartines Python bibliotekas interneto aplikacijai kurti.

Šiuo metu Korėjos dramos tampa vis populiaresnės, tačiau nepavyko surasti DI paremtos rekomendavimo sistemos, kuri būtų skirta šiam TV serialų žanrui, turinčiam unikalią struktūrą ir bruožus. Korėjos dramų duomenų rinkinys buvo surinktas iš skirtingų šaltinių. Rinkinys buvo išvalytas, jam atliktas išankstinis apdorojimas. Duomenys buvo atvaizduoti ir išsaugoti neorientuotame grafe, kurio viršūnės yra Korėjos dramos. Grafo lankai parodo ryšius tarp panašių Korėjos dramų. Atstumams tarp viršūnių matuoti Adamic Adar matas. Kuo šio mato reikšmė didesnė, tuo artimesnės viršūnės.

1 INTRODUCTION

As the number of internet users is already colossal, consequently there is no wonder that the amount of data is also increasing at a rapid speed. It is almost impossible to imagine our lives without our modern gadgets and no one can deny the importance of developing correct tools and applications which can solve our daily problems. And even if we are gleefully taking full advantage of the benefits and opportunities of this century, on the other hand, the huge amount of data and options are generating new types of problems for the modern World Wide Web users. And one of these problems is information overload which may lead to decreased productivity, information fatigue, and even to decision paralysis. And to solve these problems, recommendation engines were created and one of their types is movie recommendation engine, which has appeared because of the vast number of movies available on numerous kinds of streaming platforms and networks, in various genres and languages. It is obvious that for our brain it is impossible to filter such an amount of information and to make correct decisions. Fortunately, there are plenty of recommendation engines with free access, and each of them is built with its own algorithms and approaches. And newest methods designed to solve these problems can even involve Deep learning, Reinforcement learning approaches, and more.

Even though traditional recommendation engines are still effective and widely used, their main problem is that, they were built with limited diversity and for mass audiences. However, for the fans of specific types of cinematography such as anime, Turkish series, and in this particular case Korean television dramas, there are not so many choices. The main reasons for this issue are low demand among watchers comparing to the mass audiences, lack of datasets for training, and a need for finding an appropriate approach. For this reason, my first step before creating a model starts with data gathering, storing, cleaning, and preprocessing.

In any type of machine learning model, the quality of data plays a key role, considering that the usage of divergent and high-quality data can increase the accuracy, efficacy, and user satisfaction of recommendation engines. For this purpose, there exist several ways for data gathering like using data collection tools, APIs, surveys and questionaries, or just finding relevant datasets from open public datasets. In my case, I will use the web scrapping method using Python programming language, because the data I require did not exist in public datasets and there were no other ways to collect them. When our data will be collected and stored in a CSV file, several steps of data preprocessing actions will be done which will also be described in the later chapters.

The most important part of the work will be a creation of a recommendation model and this work starts by filtering out the data and choosing the correct features with matching characteristics [1]. And the main types of these filtering methods include collaborative filtering, content-based filtering, context-based filtering, and hybrid filtering. Each of these methods has its

strengths and limitations, and choosing the correct filtering system which can satisfy our requirements is very crucial.

Since the I‘ll take a k-dramas dataset which will contain only movie metadata with description, I will focus on creating a content-based recommender system because of the independence from user data. This filtering method mostly relies on the content, features, and attributes of the items based on the user's past preference. Unlike the collaborative method which suffers from a cold start problem that appears when there is no input information about the user or information is not enough for clustering [2], content-based filtering is still able to make good recommendations based on characteristics of movies by themselves and can suggest diverse and explainable recommendations. Of course, this method also has its own drawbacks like the inability of making occasional recommendations other than the user's last preference or capturing more subtle user preferences, but our task is also to try to overcome this issue.

To create a model based on this strategy, first we will need to understand what similarity means and this is when we will use a machine learning model. Unlike other machine learning models that are used for finding similarities between movies content, our recommendation model instead of taking only similarities of movies as only one feature, will be designed as a graph representation model and will take in an account other features and attributes also.

In graph representation, each movie will be saved as a node in space with various kinds of edges which represent features and stored in different file. And after the distance between these nodes (or in other words movies) are measured by the Adamic Adam measure which is used to calculate the closeness of two nodes based on their common neighbor. The reason why I used this model is that, unlike other traditional recommendation methods based on the content- based filtering technique, this model analyses more attributes from the dataset such as description, cast or tag and gives more advanced and clear explanation to the users.

These attributes will be taken as edges while drawing the main graph. When Adamic Adar coefficients measured between the query and each movie in the dataset, the highest five will be chosen for recommendation as the most appropriate and similar.

After the model is completely created our last goal will be to integrate a created model into our web app which will be developed by Python open-source library - Streamlit with several lines of codes. I will test the web app in the localhost, but for case, the app also can be deployed in any cloud services or hosting servers to make it alive for other internet users.

As the conclusion, in our web system users can just query the name of a Korean drama they like before from the given dataset and based on the entered k-drama’s content description and metadata, new five dramas will be recommended. There will be no need to login in, enter extra information, or collect information about the user such as age, gender, demographic situation etc.

But just from the short information about the user's last preference, our integrated, AI-powered model still will be able to suggest high-quality recommendations with transparent explanation with the ability for discovering.

2 ANALYSIS OF LITERATURE, TOPIC, AND METHODS

2.1 Literature analysis

The most common traditional filtering methods such as collaborative and content-based filtering were compared in the work of Abdul Aziz and Mubashra Fayyaz [3]. According to them, the main advantages of content-based approaches are:

- Not necessary to have a big user dataset in order to have a high accuracy
- Can handle new users when there is not enough data about them and still suggests good recommendations
- Explainable to users why curtain recommendations were generated
- New items can be commending instantly if there is access to item contents and descriptions

And comparing to other techniques and considering our topic, goals and dataset requirements, these advantages seemed like a perfect match for our demand.

According to Ashrita Kashyap, Sunita., Sneh Srivastava, Aishwarya., Anup Jung Shah there are two types of information [4] needed to create item profile which is an important part of content - based model building:

- model of the user's preference
- past user interactions with a recommender system

And to characterize these items within the system diverse kinds of element representation algorithms could be applied in other words vectorization methods. In brief explanation, vectorization is a way of converting input data usually in text form into vectors of numbers and nowadays it is widely used in NLP tasks and in AI models. In machine learning, we can also call this step - feature extraction.

The authors of "Systematic Comparison of Vectorization Methods in Classification Context" [5], in their work compared the 4 mostly used feature extraction methods such as:

1. Continuous Bag of Words (CBOW),
2. Skip-gram bag of words
3. Term Frequency–Inverse Document Frequency (TF-IDF)
4. Distributed Memory of Paragraph Vector (DM-PV)

As a result, efficiency values of all methods were almost similar, varying between 0.8-0.95, but the time these scores were achieved was different because of the complexity of the method. In conclusion, authors [5] emphasized that regardless of the vectorization method chosen the results they achieved in different aspects were very similar which led me to the idea that the

choice of vectorization technique is not so critical. For this reason, I have decided to use maybe the most widely used method for keyword extraction: the term frequency- inverse document frequency method. More about this method can be found in “method analysis” chapter.

During correct model exploration LDA model was considered as one of the options, but regarding to experiments of Sonia Bergamaschi, Laura Po and Serena Sorrentino [6], LDA didn‘t show a good performance on movie recommendation and was not able suggest the movies of the same narative and proposes incorrect entries for movies with brief plots that contain words that are also found in the target movie's storyline.

Finally, as a core idea for model, was explored the work of Abdullah Havolli, Arianit Maraj and Lorik Fetahu [7] where such model was used in Netflix movie dataset. They used undirected graph in order to show connection between nodes and edges, and main advantage of their idea is that different kinds of edges (features) can be used to show these connections. In their paper TF-IDF and Word2Vect was used to find the similarity between movies and Adamic Adam measure was used to calculate the distance between nodes. The higher the valul of Adamic coefficient the closer two nodes in the space, showing content similarity between dramas.

2.2 Topic analysis

With the rapid rise of The Korean Wave (or „Hallyu“ in Korean), caused by globalization and influencing different aspects of life such as food, culture, and style, not only in Asian countries but also all over the world [8], Korean television series, also called as Korean dramas or k-dramas, are gaining more and more popularity among watchers of different nationalities and backgrounds. To the question of why these movies are so popular, Hogarth [9] tried to answer by mentioning the Asianness, interesting storyline, beauty, high quality, a combination of traditions and modernity, and a modern picture of men and female of these dramas. And as the number of these K-dramas released each year is increasing exponentially, the number of fans is also increasing at the same speed.

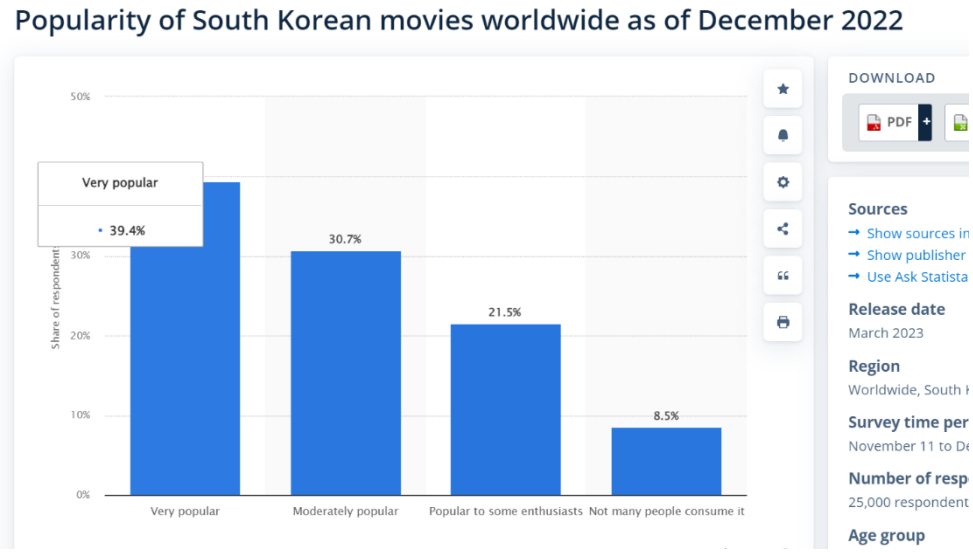


Figure 2 Popularity chart of Korean movies [23]

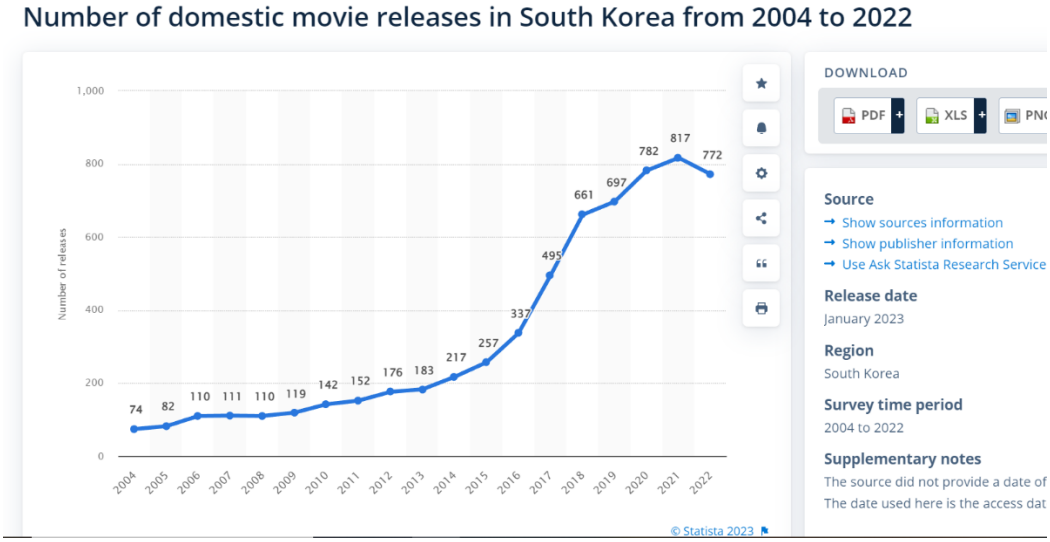


Figure 3 Number of released movies in Korea [10]

As we can notice in Figure 2, around 39% of respondents said that Korean movies were very popular in their countries, according to a survey on the global popularity of South Korean films in 2022.

Comparing to 2004, the number of Korean movies released, has also increased exponentially, by over 943 percent. (Fig 3) In recent years, Korean cinema has begun to earn notice outside, most notably because of the Bong Joon-ho's Oscar-winning film "Parasite".

Even the biggest streaming platform Netflix has started broadcasting Korean series and movies on their platform starting from 2019 and now they can even compete with other series for the top places in among Top popular non-english Shows (Fig 5) as well as in Global Chart (Fig 4)

.



Figure 4 Netflix global TV shows chart [created by author]



Figure 5 Netflix non-English TV shows chart [created by author]

However, despite their popularity, for the fans of these series, there still doesn‘t exist any recommendation websites to find which drama to watch next. Only one website I could find with the possibility to get a recommendation was *mydramalist.com*, which makes very poor recommendations based on an existing list of popular dramas without any possibility to enter your own query. As we can see (Fig 6), only one way to take recommedation is to choose from existing list of k-dramas with manually assigned recommendations.

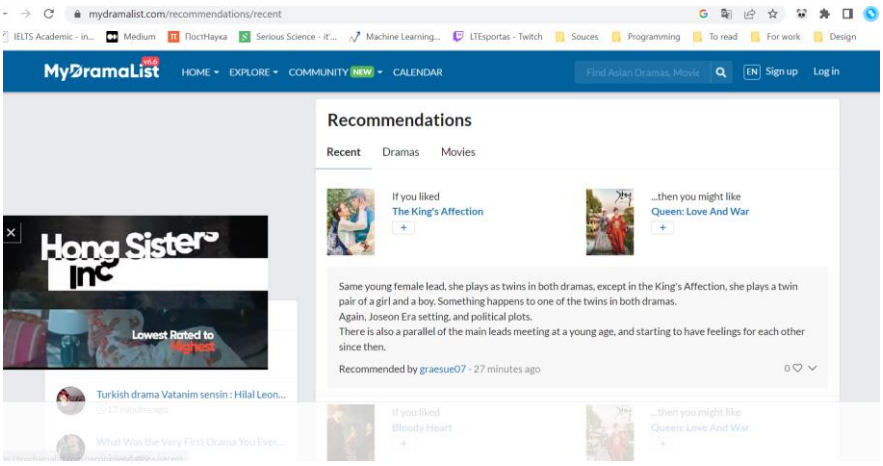


Figure 6 Screenshot of recommendation page of website [created by author]

Korean dramas are released in various genres such as friendship, romance, mystic, detective and etc. But one key moment we will consider in this work and which affected several steps of model choosing in future, is that while choosing what K-drama to watch, fans are more rely on the tags or on their own preferences such as popular screenwriters or actors, not on ratings as it can be in most of the cases. Yes, there are ratings announced by channels and people who watch them because of their popularity, but mostly it doesn't play a big role as in traditional series and movies released in english. That's why while building a model I have decided not to consider the user ratings but most attention to pay other features and attributes of K-dramas. This specific feature made an important role while choosing the correct model.

2.3 Methods analysis

2.3.1 Web scrapping

Because of several factors such as the language and culture barrier, a comparatively small amount of watchers and contributors, k-dramas datasets are not widely available. Of course, it doesn't concern the situation in the origin country where exist plenty amount of big databases of dramas and movies aired on various networks (KBS, MBC, tvN, JTBC and etc.) and streaming platforms (Viki, Viu, Kocowa, etc. and even Netflix in recent days). Each of them obviously has their own databases, but the biggest one is The Korean Movie Database (KMDb) which was launched by Korean Film Archive in 2006. But as we already mentioned database content is in a foreign language and I could not use it in my work.

Of course, without data there is no recommendation model and to find a solution, I have explored some open dataset platforms such as Kaggle, Google public datasets and etc. But most of the datasets I found were small or not complete, and the biggest dataset I found in Kaggle consist of only 1278 unique rows. For this reason, another way of data gathering, web-scaping was used.

Web scraping is a way of data gathering from the internet. And according to [11] this way includes several techniques to fulfill the task:

- Simple manual copy-pasting method
- Pattern matching
- Web mining solutions
- HTTP programming for retrieving data from static and dynamic pages
- HTML parsing
- Machine learning techniques of information extracting like processor vision web page analysis

In my case, I will do HTML parsing with Python language libraries to collect necessary information from several relevant websites, of course with respect to their legacy terms by checking Service and Legal Considerations and robot.txt file. Data, scrapped by this method, will be stored in CSV file for further analysis.



Figure 7 Web scraping structure [12]

Ryan Mitchell in his book [13] gave an explanation of web scraping as the process of writing an automated program that first queries the web server, requests data from web pages (usually in the form of HTML) and parses the data to extract useful information. If we can view the data in the browser, then we can extract the essential information through our Python script.

2.3.2 Content-based filtering

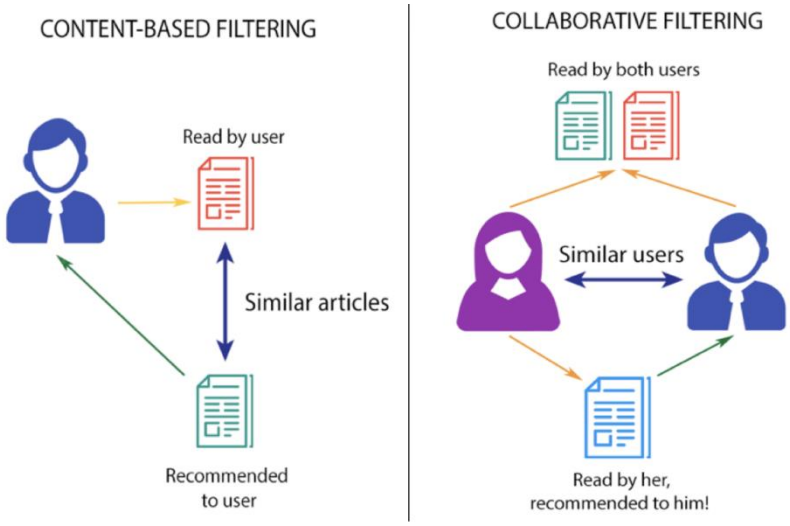


Figure 8 Action figures of CBF and CF in Recommendation systems [14]

Content-based filtering methods are based on a description and different attributes of an item and the user's last preference. [4] The core mission of such systems is to find the similarity between items. CBF is generating user preferences with respect to given data and is best suited in cases when there is no data about users but contains data about the item. In short, these algorithms will recommend items that are like those that the user liked before. And in understanding user preferences creation of the correct item profile plays a significant role.

CBF is generating user preferences with respect to given data and is best suited in cases when there is no data about users but contains data about the item. In short, these algorithms will recommend items that are like those that the user liked before. And in understanding and matching user preferences creation of the correct item profile plays a vital role. This profile includes a compilation of feature columns, attributes, and descriptions that can describe the item. The features can be in the type of numbers, strings, dates, or in the form of categorical or numeric values and are used for comparison between items to find similarities. Building a high-quality and relevant-to-topic item profile is especially important to deliver personalized recommendations and helps in matching custom items for users.

2.3.3 TF-IDF

Term frequency-inverse document frequency is the most popular keyword detection method where the significance of the word, as the name says, depends on the frequency of the word not only in the given document but also in other documents too. [15]

There are three numerical characteristics in this algorithm:

•the number of times tf(t,d), also known as term frequency, that our word t occurs in a document td;

•the total number N of documents in corpus D;

•the number of documents df(t), also known as the document frequency, which contains the given term;

$$idf(t, D) = ln\frac{N}{df(t)} \quad (1)$$

The formula of inverse document frequency is based on the above-mentioned characteristics (1).

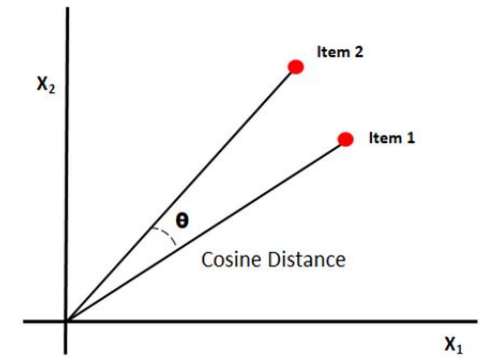
And in most of the cases this method works well and can give fairly selected keywords.

We will calculate TF-IDF matrix to consider the description while creating our graph representation model. [7]

2.3.4 Similarity measuring

There exist various text similarity metrics such as Euclidean distance, Parson correleations and maybe one of the most popular metrics is Cosine Similarity which I‘m going to use because of it‘s faster speed, low complexity and better results in catching semantics.

Cosine Similarity measures the similarity between 2 documents by measuring the cosine of angle between two vectors.



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 9 Cosine similarity in vector space and its formula [16]

As we can see from the picture above, the less the angle between vectors, as similar they are and vice versa.

2.3.5 Adamic Adar measure

This measure was introduced by Lada Adamic and Eytan Adar in 2003 [17] and defined by the sum of weights of shared neighboors between 2 nodes to determine their closeness.

$$AdamicAdar(x,y) = \sum_{u \in N(x) \cap N(y)}^{\cdot} \frac{1}{log(N(u))} \quad (2)$$

Logarithmic degree formula is introduced above (2).

Here *N(x)* and *N(y)* neighboor sets of two nodes (*x* and *y* respectively), while *u* is the common node for these nodes. The larger this value is, as close these nodes, when 0 means that nodes are not close at all.

The quantity value of $\frac{1}{log(N(u))}$ shows how important *u* in the measure.

- If x and y share a node *u* with many adjacent nodes, this node is irrelevant. If N(u) is high → 1/log(N(u)) is not high.

- If variable x and y share a node u that does not have a lot of adjacent nodes, this node is relevant. If N(u) is not high → 1/log(N(u)) is higher. [7]

2.3.6 Graph representation of network

Python is an interpreted, easy-to-learn and high-level programming language whose design principles underline code readability. And because in this work I will use this programming language only for: web scrapping, model building, and even for web application creation, the graph representation of our dataset I will also design using the open-source libraries of this language. For this purpose of complex network creation, manipulation, and structure research - Networkx python package will be used and the Pyvis library will help us to integrate a dynamic visualization of our subgraph into our web application. [18]

As main advantages of Networkx, we can mention that:

- it is all based in Python
- easy to get started but still powerful
- designed to handle the problems of modern datasets
- very flexible graph implementation

However because the graph generated by Networkx is static, and our graph is complex network, it's difficult to discover the relationships between nodes, that’s why dynamic version of the subgraph will be implemented in our web application in order to give the user a transparent view to the recommendation model with the help of Pyvis Python package.

Pyvis was developed as a wrapper for the popular JavaScript library - *visJS* , and it helps us to make our application more interactive to show the users relationships between entities with interactive canvas.

As I already mentioned in the introduction section, main reason why I chose graph representation method and measuring the nodes within it by Adamic Adar measure is that, it allows us to consider different attributes of dataset such as network, tags, genres and others, while making decisions and helps to user to discover the recommendation by themselves with integrated dynamic graph representation.

3 RECOMMENDATION ENGINE MODELLING

3.1 Architecture

Our content-based recommendation engine works in following way:

- User provides the name of the k-drama he liked before
- Web app should recommend 5 k-dramas user would like
- Along with recommendation dynamic subgraph will be drawn in web app
- User can discover the connections between recommended items and explore them

Our recommendation model will be built by using Adamic Adar measure and Cosine similarity together with TF-IDF vector space model. Adamic Adar measure will be measured on graph which will be drawn by connecting essential attributes such as tags, genre,cast and etc and relationship between them in form of nodes and edges. After that, using cosine similarity we will find „n“ similar by descriptions content movies for the given node, and will connect related movies with selected movie‘s node with edge called „SIM()“. In this case will will take into account not only various features of the k-dramas, but also the description similarity. This node will also be considered while measuring distances.

Technically it would be also possible to use only cosine similarity to generate recommendation only based on description, but by combining Adamic Adar measure and Cosine similarity with TF-IDF, we can achieve the better and more explanable results.

Measures between all nodes (k-dramas) will not be pre-calculated [7] which means that the list of recommendation movies will not be determined in advance.

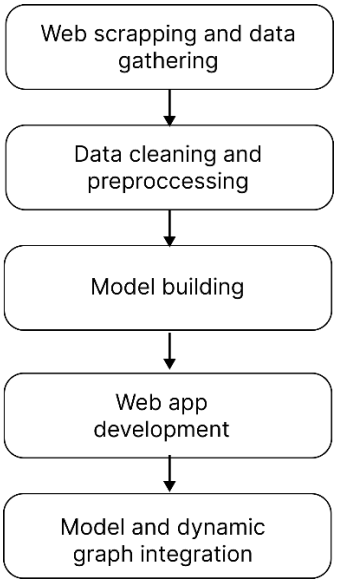


Figure 10 System building architecture [created by author]

3.2 Data gathering and web scrapping

The quality of recommendation model highly depends on the quality of data that the model trained from. And as we already defined our objectives and determined what kind of information we need to find out, the next step of data gathering is to explore from which reliable sources we will collect our dataset. Firstly, of course, I explored different public dataset platforms such as Google Dataset Search, Kaggle, GitHub and others, but only two datasets related to our topic were found: "Korean drama list "(https://www.kaggle.com/datasets/iphigeniebera/korean-drama-list-about-740-unique-dramas) containing 1278 unique rows and "Top 250 Korean dramas" (https://www.kaggle.com/datasets/ahbab911/top-250-korean-dramas-kdrama-dataset) containing consequently 250 the most popular dramas. As a result, I have decided to take the biggest dataset among them which was collected from the previously mentioned websites mydramalist.com, and from IMBD dataset, and wikipedia.com. The dataset consists of nine features and 1278 rows, but to expand it I will collect my own dataset using web scraping.

Web scrapping is a process of retrieving unstructured data from open-source websites and storing them in a structured format for further operations. And it consists of several steps [13] to be completed:

- Retrieving HTML data from website domain
- Parsing target information from the data
- Storing the information into database or any files
- In case, to move another page to repeat the process.

From my research, only one relevant website was found with desired information which is koreandrama.org, and below in the screenshot you can see how the information about one drama looks like:

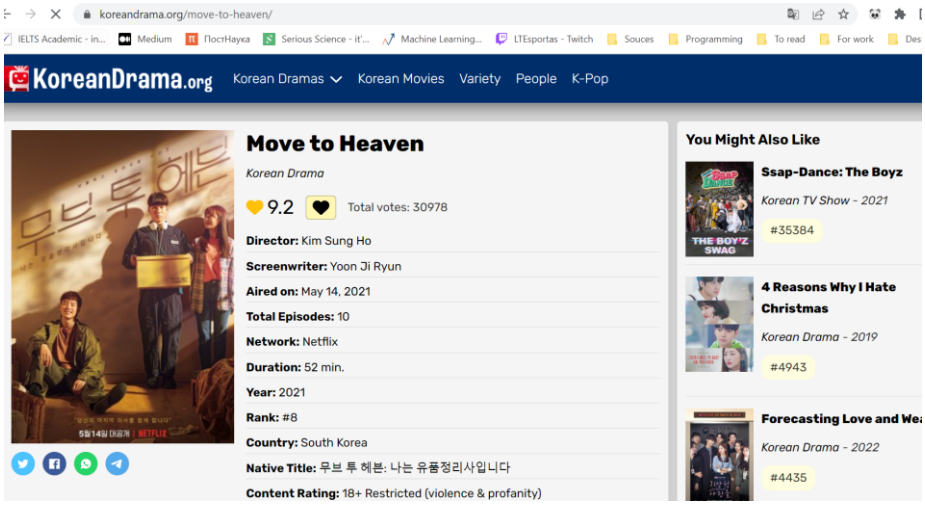
- 

Figure 11 Screenshot of separate drama page with metadata in it [created by author]

As we can notice the diversity of metadata is good enough and compatible with our existing dataset taken from Kaggle. Before starting to crawl, I checked robots.txt file to check if it is legal to crawl the indicated website or not, and here is the content of file:

```
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
Sitemap: https://www.koreandrama.org/sitemap_index.xml
```

Here "user-agent" value equals to "*" means that everyone is allowed to crawl without any restriction and there should be no issues with data privacy regulation and ethics.

The next step is writing a data extraction script in Python for web scraping and I used the BeautifulSoup package for this purpose. I wrote my code for crawling in Jupyter Notebook - a web-based tool for interactive computing.

Algorithm of web scraping process:

1. ***Importing necessary libraries.*** If some of packages do not exist, we always can install them with pip package installer. The main packages used:

– Request - sends HTTP request to the web server and returns Response Object

– BeautifulSoup – for extracting data from HTML and XML files

– Sleep – to slow down the pull request to not to overload servers

2. ***Gathering links of each k-drama page from the main page***. String methods were used to extract the text from HTML file. Also, in this part HTML file was explored to find the necessary tags and elements. But because this file was messy and difficult to read and explore, online HTML formatter tool (https://webformatter.com/html) was used to clean the file automatically. After correct tags were found and links were extracted all k-drama page links were saved into one Python set(). The compilation time took 27 minutes to complete.

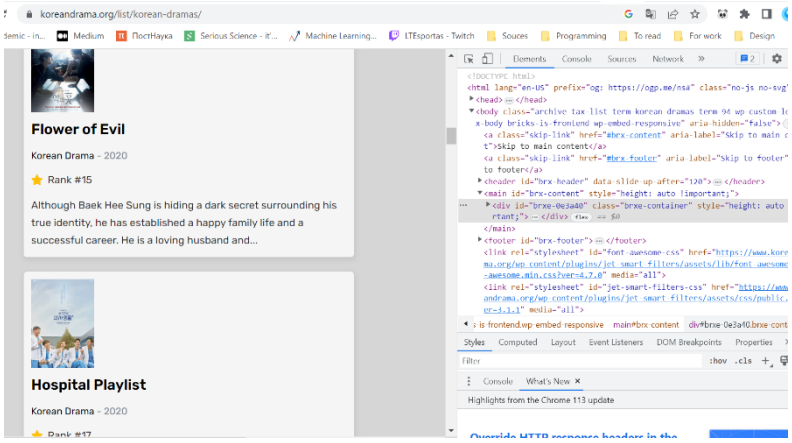


Figure 12 Main page of website with opened developers tool

3. Scraping each page to collect the metadata of each drama

Now when we have the links of all pages, we can start scraping each page one by one, to collect a metadata of each k-drama and to save into the list of lists. To find the all necessary tags regular expressions were used.

4. Creation of data from the list of lists

To easily manipulate the dataset and make analysis we will convert data saved in list of lists to Pandas dataframe objects. After some feature adjusting steps, I merged this own collected dataset with the one I found in Kaggle.

	title	image_URL	@type	description	datePublished	tags	genre	rating	extra_info	cast	director	year	du
0	A Gentleman's Dignity	https://www.koreandrama.org/wp-content/uploads...	TVSeries	Four men in their forties go through love, bre...	2022-01-02	[Adult Romance, Age Gap [Drama Life], Age Gap ...	[Comedy]	8.3	{'Director:': ' Kim Jung Hyun, Shin Woo Cheol,...	[Yoon Jin Yi, Kim Soo Ro, Yoon Se Ah, Kim Min ...	Kim Jung Hyun, Shin Woo Cheol, Kwon Hyuk Chan	2012	
1	Kiss Sixth Sense	https://www.koreandrama.org/wp-content/uploads...	TVSeries	Hong Ye Sool, the best account executive on Pl...	2021-08-26	[Boss-Employee Relationship, Ex-Boyfriend Come...	[Drama]	8.1	{'Director:': ' Nam Ki Hoon', 'Screenwriter:':...	[Yoon Kye Sang, Kim Ji Suk, Seo Ji Hye, Jin Su...	Nam Ki Hoon	2022	
2	Melancholia	https://www.koreandrama.org/wp-content/uploads...	TVSeries	A sexual scandal between a math teacher and a ...	2021-07-25	[Age Gap [Drama Life], Childhood Trauma, Corru...	[Drama]	7.7	{'Director:': ' Kim Sang Hyub', 'Screenwriter:...	[Choi Dae Hoon, Lee Do Hyun, Jin Kyung, Im Soo...	Kim Sang Hyub	2021	
3	The Barefoot Youth	https://www.koreandrama.org/wp-content/uploads...	TVSeries	Tales of the hardships and love.	2022-01-16	[]	[Drama]	7.2	{'Director:': ' Kim Yong Kyu', 'Aired on:': ' ...	[Park Geun Hyung, Go So Young, Bae Yong Joon, ...	Kim Yong Kyu	1998	
										[Shin			

Figure 13 Final dataframe view of merged dataset

In overall merged dataset with unwrapped "extra_info" column with dictionary type of values, and divided into another several columns, final dataset consisted of 7500 unique rows and fourteen features such as:

1. **Title:** str
2. **Image_URL**: URL address of poster image, we use it on our web application.
3. **Type:** str
4. **Description:** str
5. **Genre:** str
6. **Rating:** float
7. **Tags:** list
8. **Year** : float
9. **Network**: TV network name a drama was first aired on
10. **Episodes:** int

11. **Duration:** str
12. **Cast:** list
13. **Director:** list
14. **Screenwriter:** list

All the source codes can be found in the link provided in Appendix I.

3.3 Data preprocessing

First, our data must be cleaned up *(data cleaning*) by removing duplicate values that appeared after merging our scrapped dataset with the dataset from Kaggle . (*data integration*) Next we join two columns “director” and “screenwriter” into one column “director&screenwriter” to compensate the null values in the dataset (*data transformation*). During feature selection step, we will select only the column with informative and useful values and columns with unnecessary features and most amount of null values will be dropped from the dataset, such as “duration”, ”director” and “screenwriter”. “Year” column type was converted from float to datatime. After, we apply strip function into columns "director&screenwriter," "genre", "cast," and "tags" to convert their elements into list type and in places where cell element is NaN, it was replaced with an empty list "[]". After all previously mentioned steps our dataset decreased columns amount from 14 to 12 . (*data reduction*) All steps of data pre-processing can be found in the picture below.

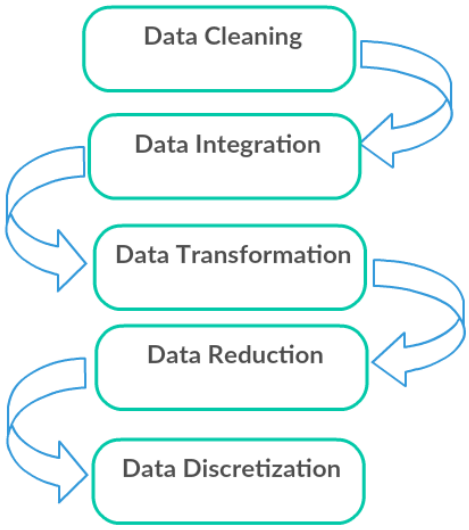


Figure 14 Data pre-preprocessing steps [19]

3.4 Exploratory data analysis

To understand the dataset and more specifically the topic itself, and to compare the k-dramas dataset with other existing datasets to see the relationships between them I will make some EDA (exploratory data analysis)

First let us use the LDA clustering algorithm to extract the topics from the k-dramas dataset and compare these topics with Netflix series database topics (all the links to used datasets can be found in Appendix II). LDA (Latent Dirichlet Allocation) is a probabilistic topic modeling composed by term by document probability distribution [6]. After the topics were extracted word cloud object was created to see the visualized picture of text-represented data.



Figure 15 Topics frequency in k-drama dataset



Figure 16 Topics frequency in Netflix series dataset

The size of words in the pictures represents the priority and frequency of the word in data, and as we can see the most frequent topic for both datasets is the same - "life", which makes sense. And while the next frequent topics are “become” for k-dramas and “find” for Netflix series, which symbolizes action verbs (which is interesting because it shows some kind of different cultural representation of the goal of the life) ,what is also interesting is that according to the next frequent topic, if the most of the movies in Netflix dataset are about “family” , for k-dramas it is about “love”.

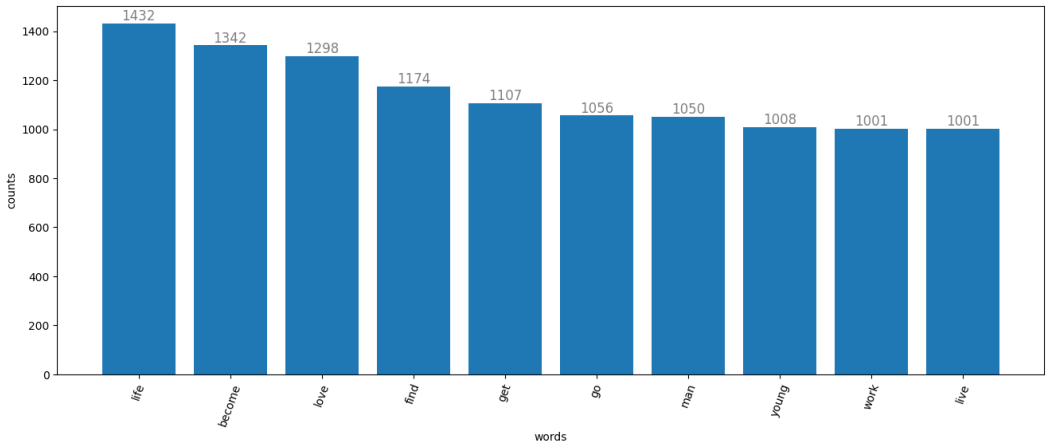


Figure 17 K-drama dataset topics distribution chart

3.5 Model building

Since we have all the essential prerequisites to start constructing the model like cleaned, preprocessed, and analyzed dataset, domain knowledge about the topic and modeling approach, we already can start the main coding. For this, I will use Python programming language and its libraries such as Pandas, Numpy, Sklearn, Networkx, and Time. [20]

First, in order to consider descriptions in our graph space model,we need to vectorize our k-dramas descriptions data by converting raw text into TF-IDF matrix and build Cosine Similarity function. To convert text into feature vectors that may be used as input to transform text to a vector of token frequencies, we import TfidfVectorizer from the package sklearn.feature_extraction.text.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.cluster import MiniBatchKMeans

# Build the tfidf matrix with the descriptions
start_time =time.time()
text_content = data_df['description']
vector = TfidfVectorizer(max_df=0.4,         # drop words that occur in more than X percent of documents
                         min_df=1,      # only use words that appear at least X times
                         stop_words='english', # remove stop words
                         lowercase=True, # Convert everything to lower case
                         use_idf=True,   # Use idf
                         norm=u'l2',     # Normalization
                         smooth_idf=True # Prevents divide-by-zero errors
                        )
tfidf = vector.fit_transform(text_content)

```

Figure 18 TF-IDF vectorization code with explanations

Here each token is assigned a number by TF-IDF that indicates how important it is to the text. As a result, texts with similar tokens will have analogous vectors. We used this approach to find cosine distance similarity in the next step.

The TF-IDF results are used to compare the similarities across k-dramas, and the cosine similarity score will be obtained by computing the dot product. To do this using Python we will use cosine_similarity function from the scikit-learn package.

```
# Find similar : get the top_n movies with description similar to the target description
def find_similar(tfidf_matrix, index, top_n = 5):
    cosine_similarities = linear_kernel(tfidf_matrix[index:index+1], tfidf_matrix).flatten()
    related_docs_indices = [i for i in cosine_similarities.argsort()[::-1] if i != index]
    return [index for index in related_docs_indices][0:top_n]
```

Figure 19 Function for measuring cosine similarity distance between vectors

Mathematically the function above produces the five vectors that closest to any supplied vector. After we can add these five vectors as a connected node to the root node. And these vectors will be connected between as one of the edges.

Now we have calcuated similarities measuring function built with cosine similarity and TF-IDF , and based on description , we can start building our indirected graph , which represents a set of nodes and a set of links between the nodes by edges, with the help of Networkx Python library for complex networks.

```
G = nx.Graph(label="MOVIE")
start_time = time.time()
for i, rowi in data_df.iterrows():
    if (i%1000==0):
        print(" iter {} -- {} seconds --".format(i,time.time() - start_time))
    G.add_node(rowi['title'],key=rowi['ID'],label="MOVIE",mtype=rowi['@type'],rating=rowi['rating'],episodes=rowi["total_epi
    for element in rowi['cast']:
        G.add_node(element,label="PERSON")
        G.add_edge(rowi['title'], element, label="ACTED_IN")
    for element in rowi['genre']:
        G.add_node(element,label="CAT")
        G.add_edge(rowi['title'], element, label="GEN_IN")
    for element in rowi['director_writer']:
        G.add_node(element,label="PERSON")
        G.add_edge(rowi['title'], element, label="DIRECTED")
    for element in rowi['tags']:
        G.add_node(element,label="TAG")
        G.add_edge(rowi['title'], element, label="TAG_IN")

    indices = find_similar(tfidf, i, top_n = 5)

    snode="Sim("+rowi['title'][:15].strip()+")"
    G.add_node(snode,label="SIMILAR")
    G.add_edge(rowi['title'], snode, label="SIMILARITY")
    for element in indices:
        G.add_edge(snode, data_df['title'].loc[element], label="SIMILARITY")
print(" finish -- {} seconds --".format(time.time() - start_time))
```

```
iter 0 -- 0.019240617752075195 seconds --
iter 1000 -- 10.902709484100342 seconds --
iter 2000 -- 22.13433313369751 seconds --
iter 3000 -- 32.89536643028259 seconds --
iter 4000 -- 43.6849262714386 seconds --
iter 5000 -- 54.58056092262268 seconds --
iter 6000 -- 65.46533226966858 seconds --
iter 7000 -- 76.24572563171387 seconds --
finish -- 81.61070489883423 seconds --
```

Figure 20 Networkx graph building function

The **nodes** are:

- Title
- Person (actor or director/screenwriter)
- Genres
- Tags

- Sim(title) – 5 similar by description content k-dramas

And the **edges** are:

- GEN_IN : relation between a genre and a k-drama
- ACTED_IN : relation between an actor and a k-drama
- DIRECTED : relation between a director/screenwriter and a k-drama
- SIMILARITY in the sense of the description
- TAG_IN: relation between tags and a k-drama

When our big graph G is built,with the help of additional function (Appendix III) we can draw a subgraph of the main graph G and let‘s see the relationship between two k-dramas :

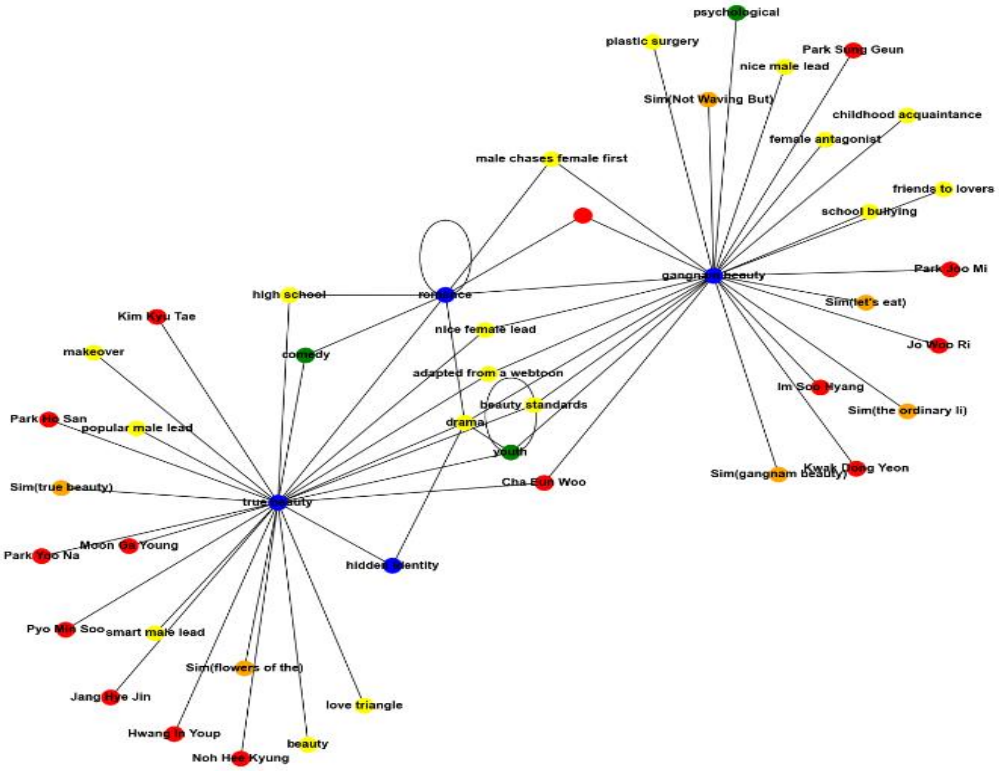


Figure 21 Subgraph of relations between 2 k-dramas

As I already mentioned even if I have created the graph in advance to save compilation time, all the distances between nodes are not going to be pre-calculated. The Adamic Adar measures between movies, based on their shared neighbors, will be calculated in a real time when user enters input query.

The last thing left in building our model is to create recommendation model using Adamic Adar measure which will be calculated by previously showed formula.

```
**************************************************
 Recommendation for 'True Beauty'
**************************************
gangnam beauty        2.249120
bel ami               2.098805
she was pretty        1.887073
the sound of magic    1.773985
queen of the ring     1.584542
dtype: float64
**************************************************
 Recommendation for '18 again'
**************************************
the great show                  7.371191
Our Summer Poem                 6.263122
ms. temper &amp;amp; nam jung gi    4.314887
my fellow citizens!             3.826498
when my love blooms             3.241930
dtype: float64
**************************************************
```

Figure 22 The similarity table of 2 dramas with their recommendations

From the screenshot picture above we can see the simiarity measure between input k-drama and 5 recommended dramas. And in the picture below that‘s how these recommendations are connected:

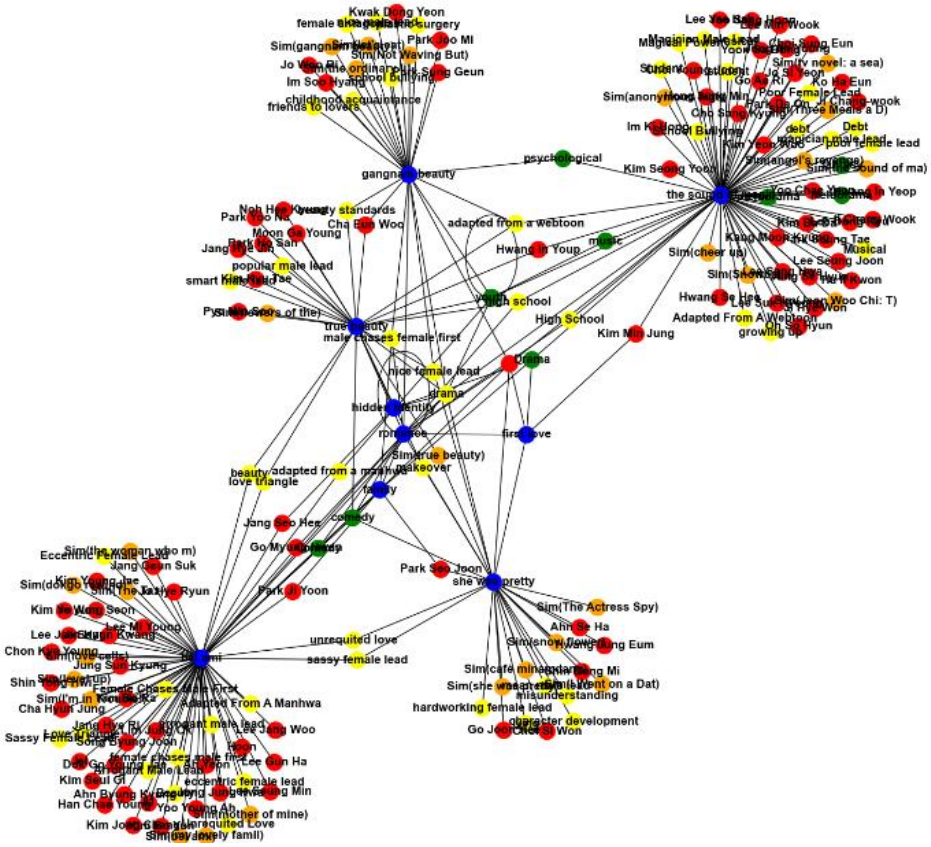


Figure 23 Subgraph of input k-drama and recommended dramas

From the subraph above we can explore the neighborhood of the target k-dramas, such as common tags, genres, the list of actors, directors and screenwriter and also another k-drama nodes similar by description content (estimated by cosine similarity) which affected the final calculations of Adamic Adar measure table.

We can say that our recommendation model works better than simple Cosine Similarity model, because before recommending the five movies to the user, the model analyses more attributes from the dataset such as tags, genres, similarity based on description and persons.

At the end of model building process, we need to save our graph in a pickle dump after processing the results for use in creating the app later. We will use Pickle package (used for serializing and de-serializing python object structures) in Python for this purpose.

3.6 Web application building

In order to create private website interface I have decided to use Python library called Streamlit.

Streamlit is an open-source Pythom framework specilly designed for machine learning engineers and it makes easy to create and share machine learning and data science projects with others. The library allows us to create and deploy such data science projects with several lines of code and in short time without any prior knowledge of web development.

Streamlit is also compatible with leading Python libraries such as pandas, numpy, scikit-learn, matplotlib, Netwrokx and others, which I have used in my project.

To create the front-end part of my application I‘ll be using built-in widgets and function of streamlit package. So after streamlit library was installed in my environment, I have created the root file named app.py. In this file I‘ll write all the necessary code for web app interface designing, k-dramas recommendation function and dynamic graph visualization function.

I‘ll run the test app in my localhost, and to do so we need go to cmd from the current location and type:

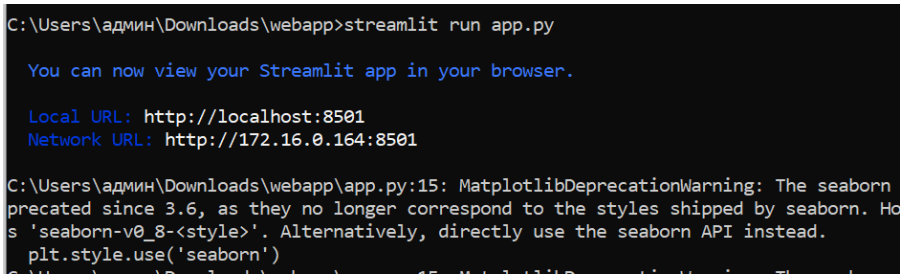


Figure 24 Cmd view of test app lunching

So my app was deployes in my localhost and I can see it from the browser. As we can see with the help of intuitive and easy-to-use capability of Streamlit library, we could create and lunch a simple designed web app in real-time with limited effort from web development side.

Before adding design elements as title, query form and button in our web page, first we need upload all necessary files for model intergation (k-drama dictionary and graph saved in pickle file format) in our directory folder.

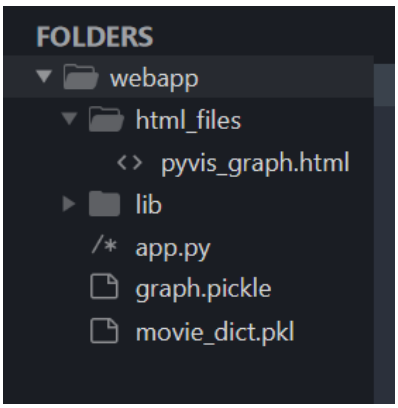


Figure 25 Directory folder of simple Streamlit project

To integrate recommendation function into the web app, first we need to load and extract our pickle files which are saved in the same directory as app.py file, and create a selectbox widget.

```
# Load the necessary python pickle files
G = pickle.load(open('graph.pickle', 'rb'))
movies_dict = pickle.load(open('movie_dict.pkl','rb'))
movies = pd.DataFrame(movies_dict)
```

Next we will pass movie names into it so that users can select the k-drama from given options. After we have created the button that recommends k-drama and recommendation function itself which will load necessary pickle files and makes recommendation, we need to fetch posters of k-dramas to be seen in the main page as well.

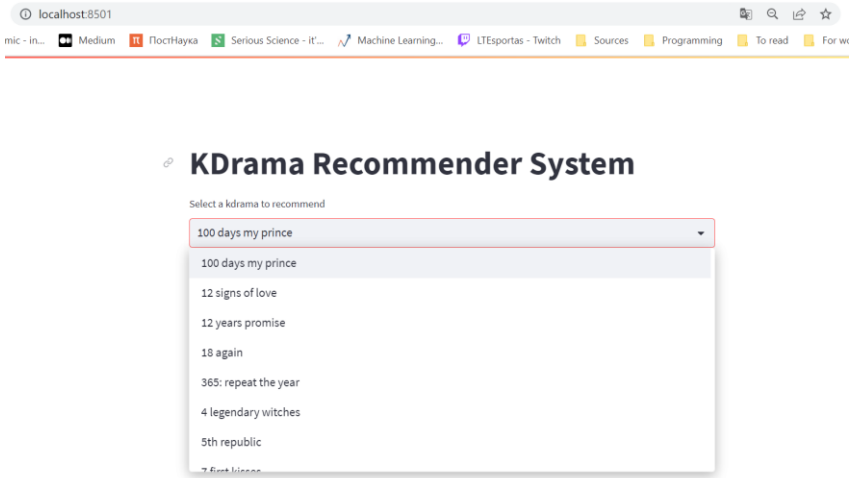


Figure 26 Main page before recommendation done

In the previous steps of data gathering I also collected each k-drama poster URL from the same website (koreandrama.org) and saved into single column called image_URL. Now it‘s time to use that column to fetch the posters and show them in our main page. Separate function was designed for this purpose and fetched posters of 5 recommended dramas are located in one row divided by 5 streamlit columns.

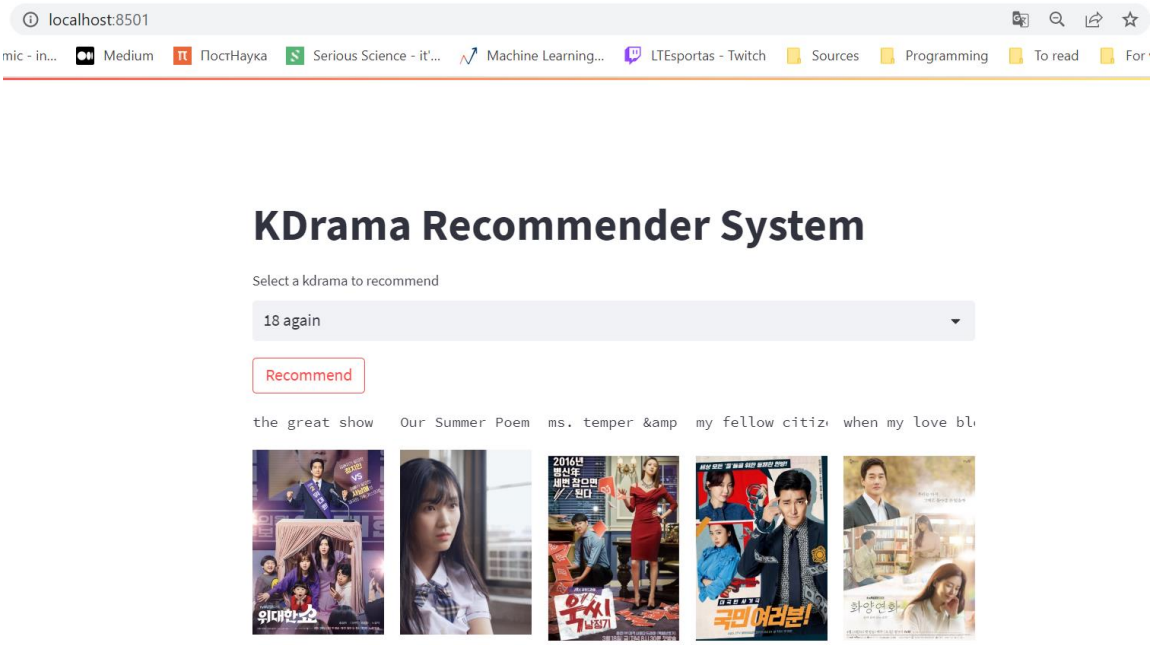


Figure 27 Web application view after recommendation

Our web app with integrated recommendation model is now ready and we can see how Streamlit is very powerful in showcasing ML models visually.

3.7 Web application improvement with integrated dynamic graph

In fact, we could leave our recommendation website like that, it has completely done and fulfills its functions excellently thanks to the integrated model and very simple but at the same time contain modern and user-friendly interface design.

But I have determined that it is a good idea to give the users the possibility to understand the relationships between an entered query and recommended dramas by themselves and provide them more transparent and explainable view of the model. In the beginning, I integrated to the app a static graph (like in the Figure 21), but because of the too many nodes and edges and the lack of possibility to zoom in, it was extremely hard to examine it. For that reason, dynamically visualized graph was implemented with the help of Pyvis- Python visualization library.

Pyvis can integrate with other complex Python network analysis libraries such as Networkx and generate JavaScript and HTML code to render the complex graphs in a web browser. And when user enters the k-drama name for recommendation, below of the recommended dramas posters, dynamic subgraph will be draw in a real time and because of it's complexity it may take some time.

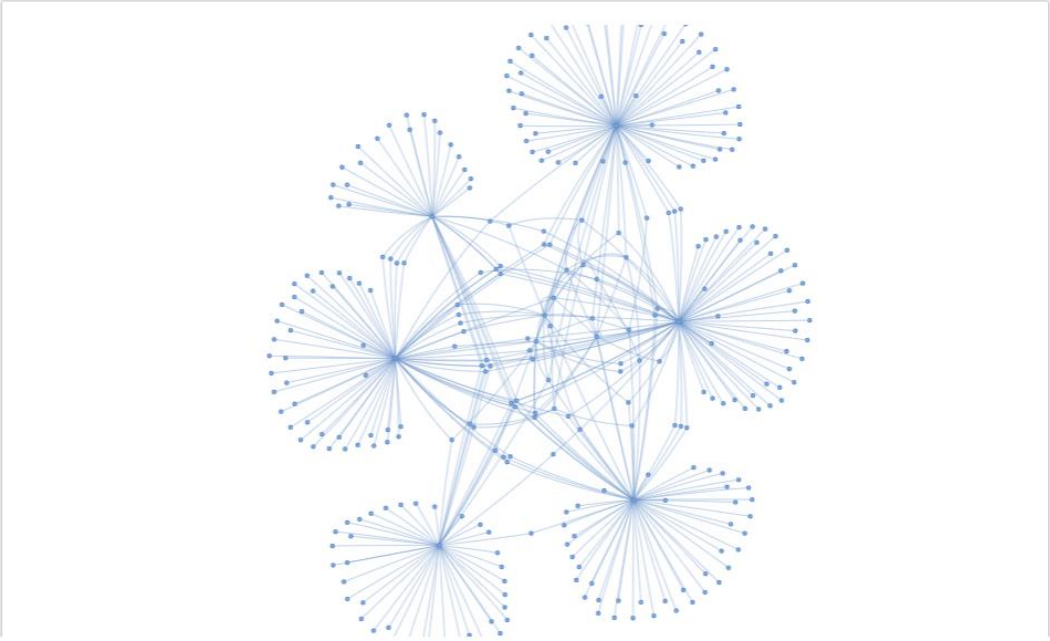


Figure 28 Whole picture of subraph

You can notice that Figure 21 and Figure 27 were generated from the same query. We also can zoom in the graph, move nodes, and interact with edges to explore particular parts:

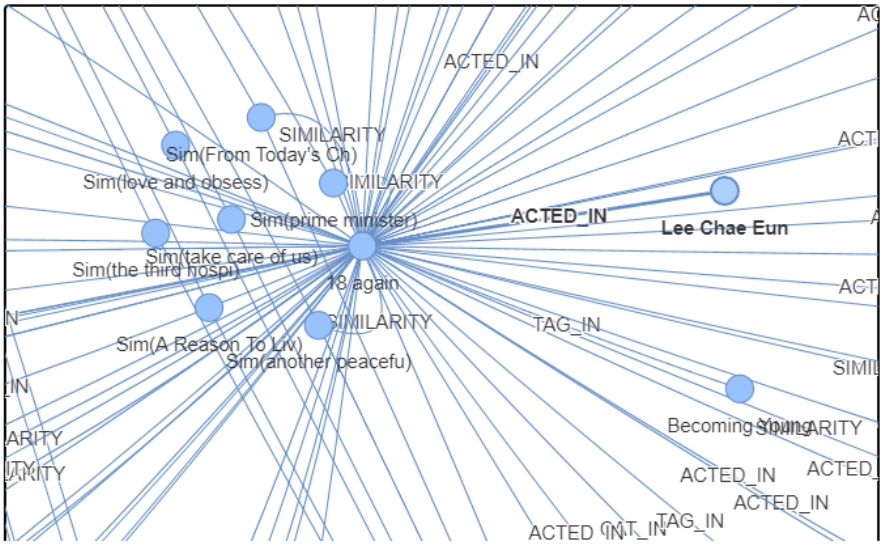


Figure 29 Zoomed part of subgraph

3.8 Testing

Testing is important part of any product development to evaluate it's performance, quality and consistency. And before testing our web application overall, because our work consists on 2 main parts, recommendation model development and web application development, we start our testing from recommendation model.

Firstly because there is no any historical test data or any users past preferences data,which is related to our topic, we can‘t test our content-based model with well known evaluation metrics for accuracy. Therefor instead of quantitive testing, I have decided to test for some qualitive features by comparing our model against another recommender (simple Cosine Similarity model).

```
1 type(recommendation('weak hero class 2'))
```

```
bullies 3
bullies
bullies 2
fatal
on your own
```

Figure 30 Recommendation with cosine similarity

```
**********************************************
 Recommendation for 'Weak hero class 2'
**************************************
Bullies                    1.505479
real:time:love season 4    1.256368
real:time:love season 2    1.256368
She's on Duty              1.181964
holy land                  1.159983
dtype: float64
```

Figure 31 Recommendation with our model

Comparing 2 different recommendations for the same query, we can see that for simple Cosine Similarity model it suggested 2 sequels of one drama (bullies , bullies 2, bullies 3), while our recommendation also suggested „bullies“ but other 4 are various. Which shows the better quality our model.

Also by discovering descriptions for some recommendations manually, it seems like they are very accurate and thanks to integrated graph we could discover them by ourselves too.

4 RESULTS AND CONCLUSIONS

4.1 Results

1. Dataset containing 7500 unique rows and 12 attributes was collected as the product of a combination of 2 other datasets, one is found in Kaggle and consists of 1278 unique values, and another one was collected by myself from koreandrama.org website using a web scraping technique and Python libraries such as BeautifulSoup and Requests. After some feature adjusting and data manipulation procedures (with the help of Pandas Python library capacities) two datasets were merged, the final data was preprocessed and prepared for further analysis in several steps like data cleaning, data integration, data transformation, and data reduction.
2. During EDA, Word Cloud of topics (extracted by the LDA model) was drawn to understand the data topics more deeply and compare with a traditional public dataset of Netflix series.
3. Before building the model itself, firstly, the similarity between description contents was found with the Cosine similarity measure and TF-IDF vectorization technique. When five similar by content dramas were found for each k-drama, the corresponding nodes were added to the main graph alongside other attribute nodes, such as tags, genres, cast, and director/screenwriter. When the main graph was designed, the graph and the k-drama dataset converted into a dictionary were dumped into a Pickle file for further use in the web app. Also, the recommendation function can calculate the distances between shared nodes (dramas) using the Adamic Adar measure to find the was created and ready to use in the main app.
4. The web app was developed by using powerful and specially designed for data science and machine learning projects Python library - Streamlit. The library also simplifies the operation of designing a web interface for our recommendation model and graph visualization.
5. When the web app was built, our model was integrated by writing corresponding codes into the main app.py file. We could lunch our web app from the command line by deploying it on our local machine, but Streamlit also allows us to deploy it on various other platforms.
6. At last, our app was improved by integrating interactive subgraph for entered querty and generated in HTML and Java Script, with the help of Pyvis library. This capability allows the user to explore the relationships between k-drama and five recommendations by themselves.

The final look of web application with recommended k-dramas for a particular query and loaded interactive graph located below of the drama posters:

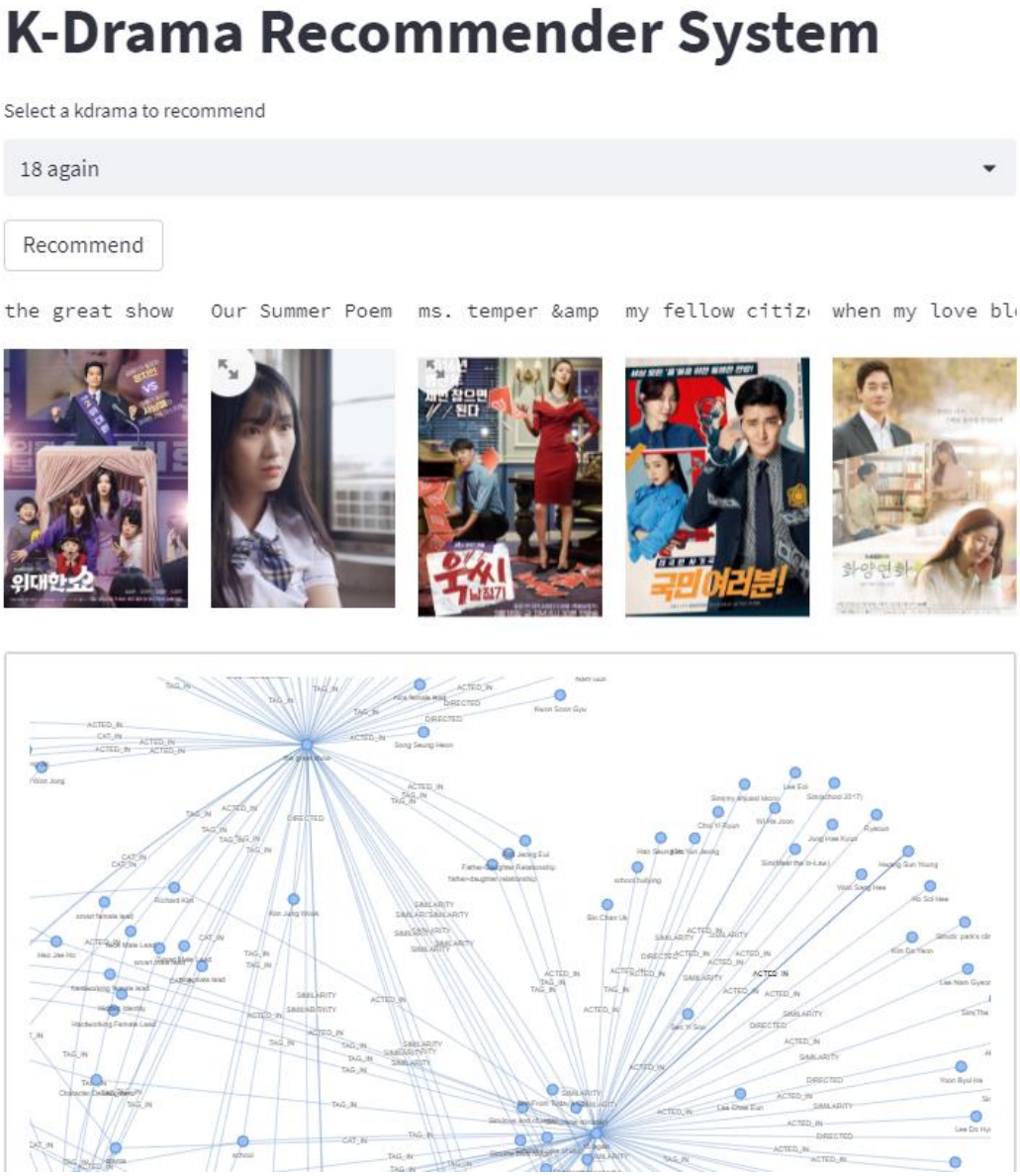


Figure 32 Final web application with integrated interactive subgraph

4.2 Conclusions

The traditional cosine similarity calculation does not consider other attributes of each movies and ignores the importance of the high-dimensional combination of attributes in the user's preference and the movie's characteristics. But our recommendation function explores the neighborhood of the target k-drama from the list of tags, genres, and directors, and can suggest broader and more precisely explored recommendations.

The final results were determined using the Adamic Adar measure, and it can highlight similarities between movies by utilizing an undirected graph to display all the relationships between nodes and show how the edges were connected.

We also discovered how straightforward it is to build simple but still robust and beautiful web apps for machine learning projects even with limited knowledge in web development. Thanks to the recent improvements in the field of web frameworks and libraries we could easily integrate our recommendation model into the web app and even improved it with the help of other libraries for better user experience.

4.3 Perspectives

In the future, it is possible to improve our model by using other vectorization techniques such as Word2Vec, which seems better than TF-IDF in terms of catching semantic meanings behind the words. Or it is also possible to use another Machine learning or even Deep Learning techniques to achieve more accurate results in finding descriptions content similarity.

It is also good to integrate additional possibilities to the web application, such as the ability to choose the number "n" of k-dramas for recommendation or to filter the results.

5 REFERENCES

[1] Sambandam Jayalakshmi, Narayanan Ganesh, Robert Cep,Janakiraman Senthil Murugan, „Movie Recommender Systems: Concepts, Methods, Challenges,,“ *MDPI,* 2022.

[2] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, Robin Burke , „Research commentary on recommendations with side information: A survey and research directions,“ *Science Direct,* t. 37, 2019.

[3] Abdul Aziz, Mubashra Fayyaz, „Comparison of Content Based and Collaborative Filtering in Recommendation Systems,“ įtraukta *Conference: International Conference on Multimedia Information Technology and Applications*, Vietnam, 2021.

[4] Ashrita Kashyap,Sunita. B, Sneh Srivastava,Aishwarya. PH, Anup Jung Shah, „A Movie Recommender System: MOVREC using Machine Learning,“ *International Journal of Engineering Science,* pp. pp. 1-6, 2020.

[5] Urszula Krzeszewska, Aneta Poniszewska-Maranda, Joanna Ochelska-Mierzejewska, „Systematic Comparison of Vectorization Methods in Classification Context,“ *MDPI,* p. 14, 2022.

[6] Sonia Bergamaschi, Laura Po and Serena Sorrentino, „Comparing Topic Models for a Movie Recommendation System,“ 2014.

[7] A. Havolli, A. Maraj and L. Fetahu, „Building a content-based recommendation engine model using Adamic Adar Measure; A Netflix case study,“ *11th Mediterranean Conference on Embedded Computing (MECO),* pp. pp. 1-8, 2022.

[8] Mutiara Ramadhani Sugiri, Viny Christanti Mawardi, Janson Hendryli, „IMPLEMENTING HYBRID FILTERING ON KOREAN DRAMA RECOMMENDATION THROUGH K NEAREST NEIGHBOR ALGORITHM,“ *Indonesian journal of interdiciplinary science,* 2022.

[9] H. K. Hogarth, „The Korean Wave: An Asian Reaction to Western-dominated Globalization,“ 2013.

[10] Number of domestic movie releases in South Korea from 2004 to 2022, Statista Research Department.

[11] Manjushree B, Sharvani G , „Survey on Web scraping technology,“ t. Volume XVI, nr. Issue VI, 2020.

[12]

[13] R. Mitchell, Web Scraping with Python: Collecting More Data from the Modern Web. Second edition, O'Reilly Media, Inc, 2018.

[14]

[15] Lukáš Havrlanta, Vladik Kreinovich, „A simple probabilistic explanation of term frequency-inversedocument frequency (tf-idf) heuristic (and variations motivated by this explanation),“ *INTERNATIONAL JOURNAL OF GENERAL SYSTEMS,* t. 46, 2017.

[16]

[17] Lada A Adamic, Eytan Adar, „Friends and neighbors on the Web,“ 2003.

[18] Aric Hagberg, Drew Conway, „NetworkX introduction: Hacking social networks using the Python programming language“.

[19]

[20] Marcus Lim,Azween Abdullah, NZ Jhanjh, „Indexed Metrics for Link Prediction in Graph,“ *(IJACSA) International Journal of Advanced Computer Science and Applications,,* t. Vol.11, 2020.

[21] Ramni Harbir Singh, Sargam Maurya, Tanisha Tripathi, Tushar Narula, Gaurav Srivastav , „Movie Recommendation System using Cosine Similarity and KNN,“ *International Journal of Engineering and Advanced Technology (IJEAT),* %1 t. iš %2Volume-9, nr. Issue-5, 2020.

[23] statista.com, Popularity of South Korean movies worldwide as of December 2022, Mar 27, 2023: Statista Research Department,, 2023.

[24] Number of domestic movie releases in South Korea from 2004 to 2022, Statista Research Department, January 2023.

6 APPENDICES

1. Complete source code can be accessed via my GitHub link: https://github.com/Satima2002/k-drama-recommendation

2. Dataset links

Korean drama list [about 1278 unique dramas]	https://www.kaggle.com/datasets/iphigeniebera/korean-drama-list-about-740-unique-dramas
Netflix series dataset	https://www.kaggle.com/datasets/shivamb/netflix-shows

6.1 Sources

https://medium.com/analytics-vidhya/content-based-movie-recommender-system-c684d663ee38

https://www.zenrows.com/blog/robots-txt-web-scraping#most-common-robots-txt-rules

https://networkx.org/documentation/networkx-1.10/reference/generated/networkx.algorithms.link_prediction.adamic_adar_index.html

https://webformatter.com/html

https://docs.streamlit.io/

https://pyvis.readthedocs.io/en/latest/

https://www.freecodecamp.org/news/web-scraping-sci-fi-movies-from-imdb-with-python/

https://neptune.ai/blog/vectorization-techniques-in-nlp-guide

https://stackoverflow.com/questions/26192727/extract-content-of-script-with-beautifulsoup

https://stackoverflow.com/questions/46326991/how-to-scrape-links-from-wikipedia-with-python