LUDWIG-MAXIMILIANS-UNIVERSITÄT AT MÜNCHEN
Department "Institut für Informatik"
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Heinrich Hussmann

Master's Thesis

Explanation Adaptation for Hybrid Multimedia Recommendation System

Zhe Li
Li.Zhe@campus.lmu.de

Bearbeitungszeitraum:	02.08.2019 to 02.03.2020
Betreuer:	Zhiwei Han and Yuanting Liu
Verantw. Hochschullehrer:	Prof. Dr. Andreas Butz

Aufgabenstellung

THESIS TITLE Explanation Adaptation for Hybrid Multimedia Recommendation System

Problem Statement How to make the explainable recommendation system interact with the user in a transparent and efficient way

Tasks Build a hybrid movie recommendation system to compare some different ways of recommendation styles and design an adaptation style for the recommendation by using an explanation template.

Keywords Recommendation System, Explainable Recommendation, Explanation Adaptation, Graph Neural Network, UMNN

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig angefertigt, alle Zitate als solche kenntlich gemacht sowie alle benutzten Quellen und Hilfsmittel angegeben habe.

München, March 02,2020 .....

Acknowledgments

First of all, I would like to appreciate my supervisor Zhiwei Han and Yuanting Liu for their valuable advice on my thesis. They gave me goals and directions for finishing my thesis.

And I would like to appreciate Prof.Andreas Butz and Prof.Heinrich Hussmann, their rigorous academic attitude and profound knowledge, the unpretentious and approachable personality charm have a profound influence on me, and it is my role model to learn.

And I would like to appreciate my parents for help during my studies, with funding and love. Finally, I would like to say thanks to all my friends who accompanied me during my master studies and thank them for all the helpful suggestions and opinions they provided for me.

Abstract

With the development of communication technology and data science, the relationship between people and information has evolved from one-way, people looking for information, to the current two-way relationship. The recommendation system plays a more important role in it. An explainable recommendation system can interact with users in a transparent and efficient way. We build a hybrid movie recommendation system to compare some different ways of recommendation styles and design an adaptation style for the recommendation by using an explanation template. And we propose a new methodology for the "adaptation rule" of recommendation explanation. A new bipartite graph-based algorithm is designed to represent our user-movie network.

Contents

List of Figures

List of Tables

List of Algorithms

Listings

1 Introduction

> That men do not learn very much from the lessons of history is the most important of all the lessons that history has to teach.
>
> Aldous Huxley

1.1 History of Recommendation System

Review the changes in the way we look for information so far since the birth of the Internet. In the earliest days, information was scarce. The scattered information led to low efficiency of searching. The main method of information transmission was that people looking for information. Later, the information has gradually been enriched. Some people or companies have gathered kinds of information on some websites, and people can search through the category navigation.

With the increasing amount of information, artificially added categories can no longer cover all information, so another type of information acquisition way - search, is born. Typical companies are Google and Baidu.

With the development of communication technology and data science, the relationship between people and information has evolved from one-way people looking for information to the current two-way relationship. People find what they need in massive amounts of information, and at the same time, massive amounts of information are also find ways to matching with the audience.

Balabanovic designed an adaptive agent for automated web browsing [Balabanovic et al., 1996], in which we can see the prototype of the original recommendation system. Such traditional recommendation systems are unsatisfactory because most of the recommendation systems we use today only make recommendations. Users do not understand the reason and meaning of recommendations. In many cases, if the recommendation algorithm is not accurate or is completely wrong, the results of the recommendation will be strange and deepen the users' doubts and distrust. Recommendations are generated by the entire black-box operations. Users cannot choose but passively accept recommendations and cannot improve the effect of recommendation through interaction.

Yongfeng et al. introduced " Explainable Recommendation " which was a new survey and perspectives for recommendation system [Zhang and Chen, 2018]. However, much of the research up to now has been descriptive in the generation of explanation. What we can do with explanations and what feedback users can give to the system about the explanation. Based on feedback on explanations, what optimizations can the recommend system make for that and how can it make such optimizations. Little research has been done on these issues.

In this paper, we compare the different ways in 5 recommendation styles, provide an overview of the relationship among recommend system, recommendation explanation, and explanation adaptation. We explore the ways in which design an adaptation style for the recommendation explanation module and feedback scoring module. And we propose a new methodology for the "adaptation rule" or "adaptation algorithm" of recommendation explanation.

From this evolutionary process, we can see that the emergence of recommendation systems has two important prerequisites, one is information overload, and the other is ambiguous requirements. With the rapid development of today's technology and the increasing amount of data, people feel more and more helpless in the face of massive data. In order to solve the problem of information overload, the computer scientists have proposed a recommendation system, which corresponds to a search engine and can also be referred to as a recommendation engine.

Search engines require people to have a clear purpose. They can turn people ?s search for information into precise keywords, then give them to the search engine and finally return them to a series of lists. Users can give feedback on these results. But it will have the problem of the Matthew effect [Merton, 1968], which will cause the more popular things to become more popular as the search process iterates, making those less popular things sink to the sea.

The recommendation engine is more suitable for people who have no clear purpose, or that their purpose is ambiguous. Generally speaking, the user does not even know what he wants. The user's historical behavior or the user's interest preferences or the user's demographic characteristics are transmitted to the recommendation system, and then the recommendation system uses some algorithm to generate a list of items that the user may be interested in. The user is passive to search engines. People only focus on high-exposure items and ignore low-exposure items, which is called the long-tail theory [Anderson and Andersson, 2004], can be used to explain the correctness and rationality of the recommendation system well. Experiments have shown that the low-exposure items in the long tail position generate no less than profit from selling only high-visibility items. The recommendation system can provide opportunities for all items to be recommended, for discovering the potential profits of long-tail projects.

When a user's needs are clear, he tends to use search. When the requirements of the user are not clear, some behaviors and preferences data based on the user's history can be used to generate a pre-recommendation, to make the user stay in the system. Some new demands may arise during the user's browsing of information.

From the above sections, we can see that the recommendation system is a multi-win-win thing:

For users, they can discover what they are interested in and improve the user experience;

For items, it is possible to discover the utilization efficiency of long-tail items and revitalize the overall resources; For the platform, it is able to capture user value and business value.

1.2 Interaction Between People and Internet

We can divide the scenarios where people interact with the Internet into three situations as shown in figure 1.1.

(a): People proactive use search engines to get information, and the Internet generates corresponding search results based on what users search for and returns the results according to some organizational structure. The volume of this information "I" is very large and complex. People read this information "I" and try to find the information "i" ($i \in I$) they need and extract it.

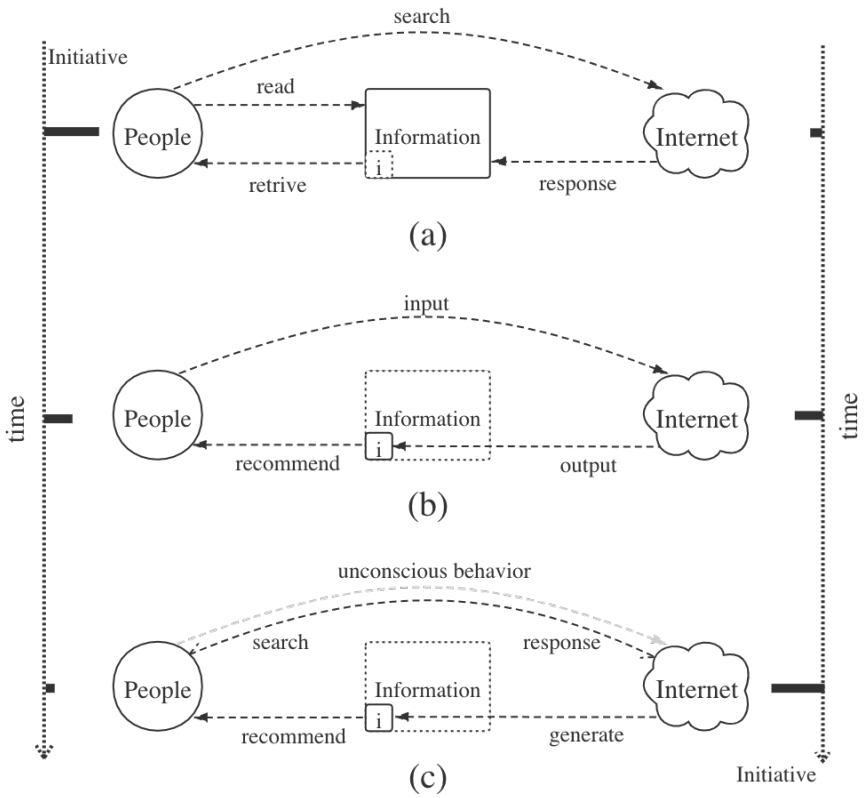
(b): People enter some information, and the Internet outputs some recommendations through a recommendation system, and selects a small part of the information that the user needs from a huge information stream, and sends it to the user.

(c): The recommendation engine actively searches for audiences that match the content it is recommending, collects data generated by users' unconscious behaviors on the Internet, and analyzes it. In addition, it will actively generate information and recommend that to the corresponding people. For example, a customer purchased a certain product x on Amazon. After a few days, he watched the video on Youtube and found that the advertisement before the video was put by the manufacturer of product x. And the advertising content was about another product y similar to product x. This is one of the ways in which (c) interaction scenarios simply appear in our lives. In the future, human interaction approaches for the recommendation engine based on the scenario (c) will be more natural and diverse.

From a vertical perspective, over time and the development of computer technology. Human initiative is decreasing, and correspondingly, Internet initiative is increasing. This may be the trend of future recommendation system development.

Internet giants such as Google and Baidu, which relied on the advantages of search engine technology in the early days, quickly grew into giant companies in the Internet industry. With the development of Internet technology and computing science today, the speed of information gener-

Figure 1.1: Interaction styles between people and Internet



ation and transmission is getting faster and faster, and the volume is getting larger and larger. The recommendation system plays a very important and irreplaceable role in the interaction between people and the Internet, as described in (b). But on the other hand, we can see in (c) that recommendation engines have great development potential to achieve the same magnitude as search engines. In this paper, we will do some valuable research and exploration in aspects such as explanation and adaptation for recommend system.

1.3 Our Contributions

In this paper, we compare the different ways in 5 recommendation styles, provide an overview of the relationship among recommend system, recommendation explanation, and explanation adaptation. We explore the ways in which design an adaptation style for the recommendation explanation module and feedback scoring module. And we propose a new methodology for the "adaptation rule" or "adaptation algorithm" of recommendation explanation.

2 Related Work

> That men do not learn very much from the lessons of history is the most important of all the lessons that history has to teach.
>
> Aldous Huxley

2.1 Recommend System

Francesco et al. systematically and comprehensively describe the recommendation system in the handbook for Recommender Systems [Ricci et al., 2011]. A recommendation system is a software tool and technology used to provide suggestions and recommendations to users. The recommendations are designed to provide users with choices in various decision-making processes, such as which products to pick, which movies to watch, or what news to read. It turns out that the recommendation system can be used to solve the problem of user information overload, and has become one of the most powerful and popular tools in e-commerce. Corresponding to the recommendation system is search engine technology, various technologies for generating recommendations have been proposed, and in the past ten years, many recommendation systems have been successfully deployed in commercial environments.

The entire recommendation system can be regarded as a processing plant, inputting user and item data, outputting a list of items that the user may be interested in, and then taking the first several from the item list as the recommendation result to the user.

In this process, we need to do some filtering and sorting. When outputting results, it is best to let users know why this is recommended, so that the user's acceptance will be higher.

Through some mathematical algorithms, guess what users might like. The recommendation algorithms use some of the user's behavior and some mathematical algorithms to infer what the user might like.

The recommendation algorithm is performed by a machine. While it can process a large amount of information, it is not so humane. In the process of pushing recommendations to the user, the powerful processing energy of the machine can deal with much more information than humans.

2.2 Explainable Recommendation

2.2.1 Definition

Explainable Recommendation refers to the personalized recommendation algorithms that address the problem of why - they not only provide users with the recommendations, but also provide explanations to make the user or system designer aware of why such items are recommended. In this way, it helps to improve the effectiveness, effeciency, persuasiveness, and user satisfaction of recommendation systems. Yongfeng et al. first highlight the position of explainable recommendation in recommender system research by categorizing recommendation problems into the 5W, i.e., what, when, who, where, and why, in their survey [Zhang and Chen, 2018].

When: time-aware recommendation
What: application-aware recommendation
Who: social recommendation
Where: location-based recommendation
Why: explainable recommendation

Research on the question about "Why" has created a new research area: explainable recommendation.

To make personalized recommendation models intuitively understandable, researchers have more and more turned to the study of Explainable Recommendation Models, where the recommendation algorithm not only provides a recommendation list as output, but also naturally works in an explainable way and provides explanations to accompany the recommendations [McAuley et al., 2015].

There are many different display styles of explanations: Explanation based on Relevant Users or Items, Feature-based Explanation, Textual Sentence Explanations, Visual Image Explanations and Social Explanation. Explanation based on relevant users or items, which presents nearest-neighbor users or items as explanation, and the relevant users or items are provided by user-based or item-based collaborative filtering methods. In this thesis, we will use a Path Generate Algorithm based on nearest-neighbor to build the model of our explainable recommendation system.

Explanation based on relevant users or items, which presents nearest-neighbor users or items as explanation, and the relevant users or items are provided by user-based or item-based collaborative filtering methods. User-based and item-based collaborative filtering [Sarwar et al., 2001], [Cleger-Tamayo et al., 2012] are two fundamental methods for personalized recommendation.

Feature-based explanation, which provides users with the item features that match with the target user?s interest profile as explanation.

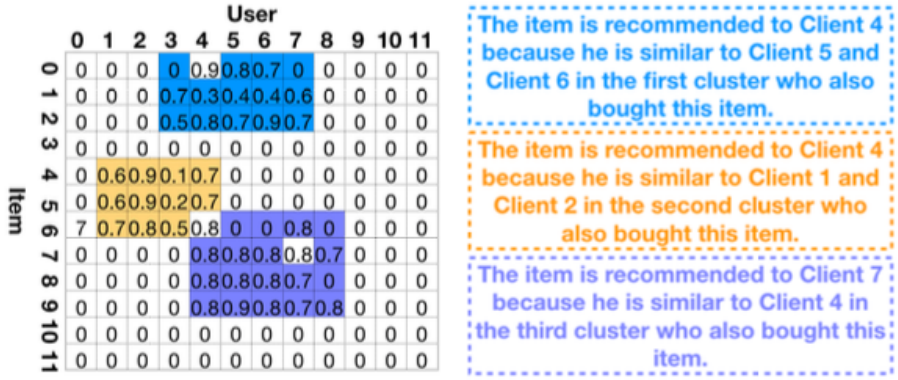
We combine these two and build a hybrid explainable recommendation system that can handle both types of explainable recommendations simultaneously.

2.2.2 Graph-based Models for Explanation

Graphs can be used to represent user-user or user-item relationships in most cases. Various studies have assessed the efficacy of Graph-based Models for Explanation. Graph learning approaches such as graph-based propagation and graph clustering can be used to generate the explainable recommendations. Most research on modeling for explainable recommendation has been carried out in Graph-based model. He et al. [He et al., 2015] described a tripartite graph to model the user-item-aspect ternary relation for a top-N recommendation. They proposed TriRank, a common-used algorithm for ranking on tripartite graphs by regularizing and normalizing the fitting constraints and smoothness.

Based on the user-item bipartite graph for the explainable recommendation, Reinhard and his team [Heckel et al., 2017] proposed to conduct over-lapping co-clustering, without using external information such as aspects. The users have similar interests and the items are of similar features in each co-cluster.

Figure 2.1: OC-uLaR algorithm example [Heckel et al., 2017]

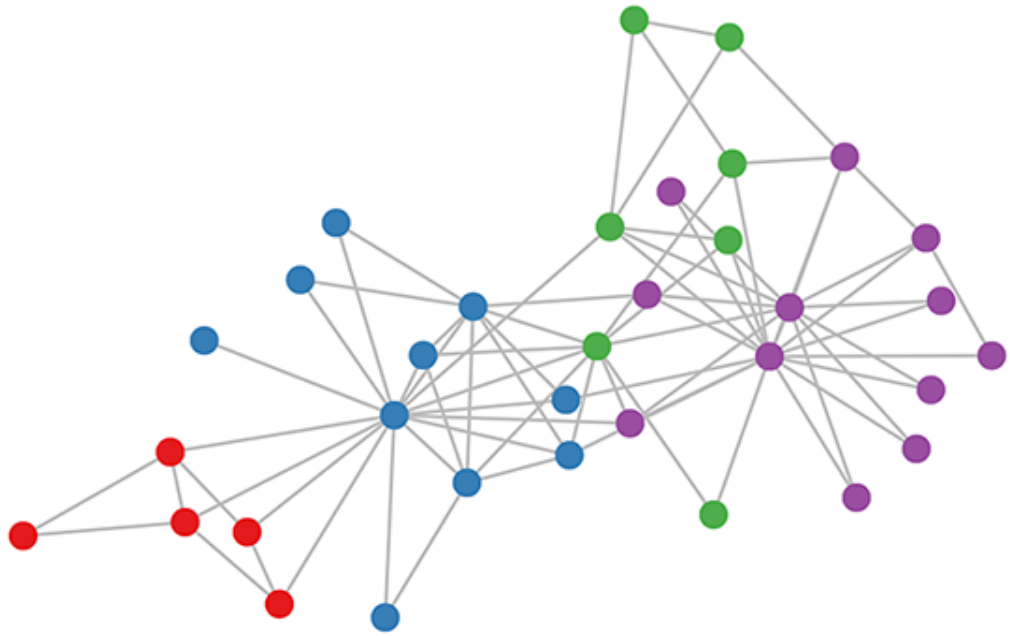


2.3 Graph Neural Network Algorithm

2.3.1 GCN

Graph Convolutional Network (GCN) can be used to build most of the large datasets we already have. For these models, the goal is then to learn a function of signals/features on a graph $G = (\upsilon, \varepsilon)$.

Figure 2.2: modularity-based clustering GCN



(i.) A feature description x_i for every node i; summarized in a N×D feature matrix X (N: number of nodes, D: number of input features).

(ii.) A representative description of the graph structure in matrix form; typically in the form of an adjacency matrix A (or some function thereof). and produces a node-level output Z (an N×F feature matrix, where F is the number of output features per node). Graph-level outputs can be modeled by introducing some form of pooling operation. [Duvenaud et al., 2015]

$$f(H^l, A) = \sigma(AH^{(l)}W^{(l)});$$

the above is a very simple form of a layer-wise propagation rule. Duvenaud and his team [Duvenaud et al., 2015] address two limitations of this simple model:

(j.) Multiplication with A means that, for every node, we sum up all the feature vectors of all neighboring nodes but not the node itself (unless there are self-loops in the graph).

(jj.) A is typically not normalized and therefore the multiplication with A will completely change the scale of the feature vectors (we can understand that by looking at the eigenvalues of A). They essentially describe a new propagation rule introduced in [Kipf and Welling, 2016]:

$$f(H^l, A) = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)});$$

2.3.2 GraphSAGE

It turns out that low-dimensional vector embeddings of nodes in large graphs are extremely useful as feature inputs for various prediction and graph analysis tasks. Hamilton and his team [Hamilton et al., 2017]describe the embedding generation, or forward propagation algorithm , which assumes that the model has already been trained and that the parameters are fixed. In particular, they assume that they have learned the parameters of K aggregator functions. The key idea of the method GraphSAGE behind their approach is that they learn how to aggregate feature information from a node's local neighborhood. Specifically, they use Embedding generation (i.e., forward propagation) algorithm in algorithm 1.

Our kernel recommend algorithm for generating path is conceptually inspired by a classic algorithm for testing graph isomorphism and GraphSAGE algorithm.

Algorithm 1: GraphSAGE Embedding Generation Algorithm

Input: Graph G(v, ε); input features {x$_v$, $\forall$ v $\in$ V}; depth K; weight matrices W^k, $\forall$k $\in$ {1, ..., K}; non-linearity σ; differentiable aggregator functions AGGREGATE$_k$, $\forall$ k $\in$ {1, ..., K}; neighborhood function N : v $\rightarrow 2^v$

Output: Vector representations z$_v$ for all $v \in$ V

$h_v^0 \leftarrow \chi_v, \forall v \in V$

for $k = 1...$ **do**

  for $v \in V$ **do**

    $h_{N_{(v)}}^k \leftarrow AGGREGATE_k(\{h_u^{k-1}, \forall u \in N(v)\})$;

    $h_v^k \leftarrow \sigma\left(W^k \cdot CONCAT(h_v^{k-1}, h_{N_{(v)}}^k)\right)$;

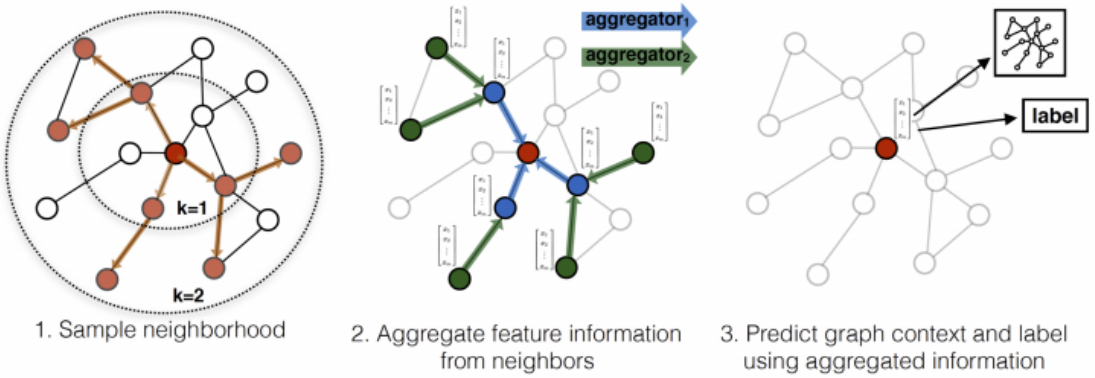
  end

  $h_v{}^k \leftarrow h_v{}^k / \left\|h_v{}^k\right\|_2, \forall v \in v$;

end

$z_v \leftarrow h_v^k, \forall v \in v$

Figure 2.3: Visual illustration of the GraphSAGE sample and aggregate approach

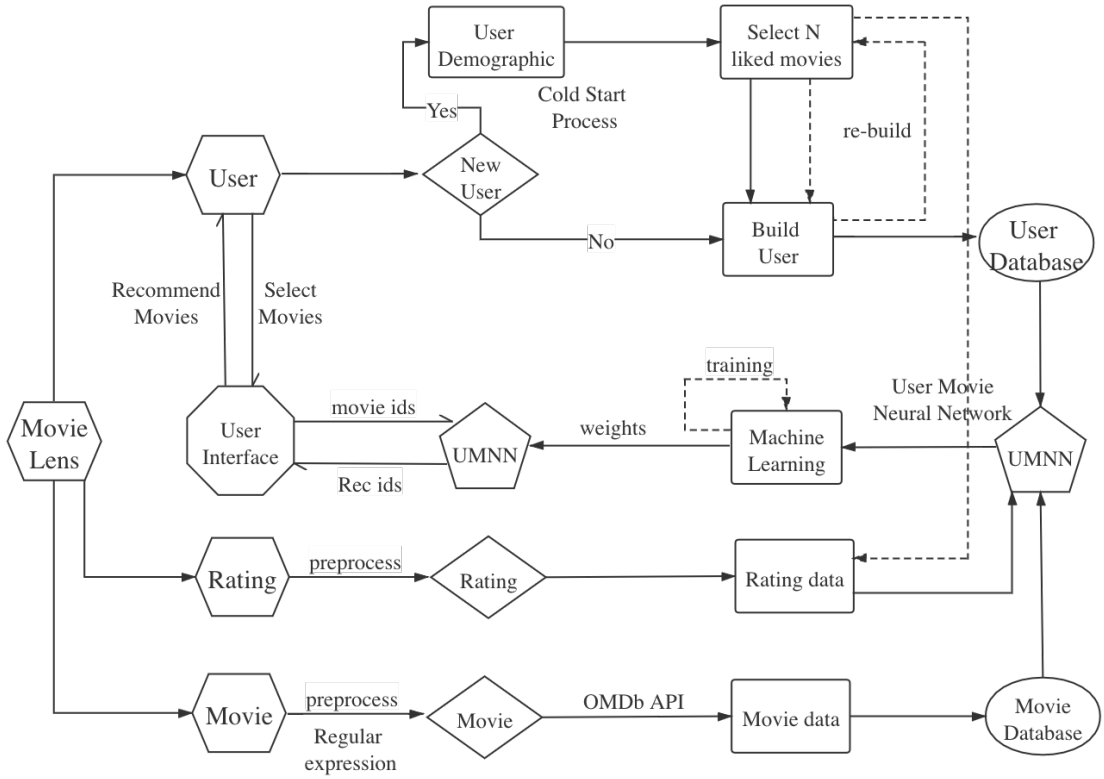


3 System Design

> That men do not learn very much from the lessons of history is the most important of all the lessons that history has to teach.
>
> Aldous Huxley

3.1 System Architecture Diagram

Figure 3.1: System Architecture



3.1.1 MovieLens Dataset

We use the MovieLens dataset from the GroupLens Research Group at the University of Minnesota. The MovieLens 1M dataset is used as the data source in this paper. It contains 1 million ratings of 4,000 movies from 6,000 users. It is divided into three tables: movie ratings, movie metadata (genre style and age), and demographic data about users (age, zip code, gender, and occupation).

3.1.2 4 Data Tracks

In the system architecture diagram. The two outputs from the MovieLens extract the movie table and rating table as the input of the movie module, and extract the user table as the input of the user module. The user and the movie, respectively, represent two paths, which represent the behavior trajectory when a user or a movie is entered in the system. This paper divides the entire

recommendation system into four parts according to the business path, which are the user data track, movie data track, rating data track, and recommendation generation track. In the following paragraphs, we introduce each track separately.

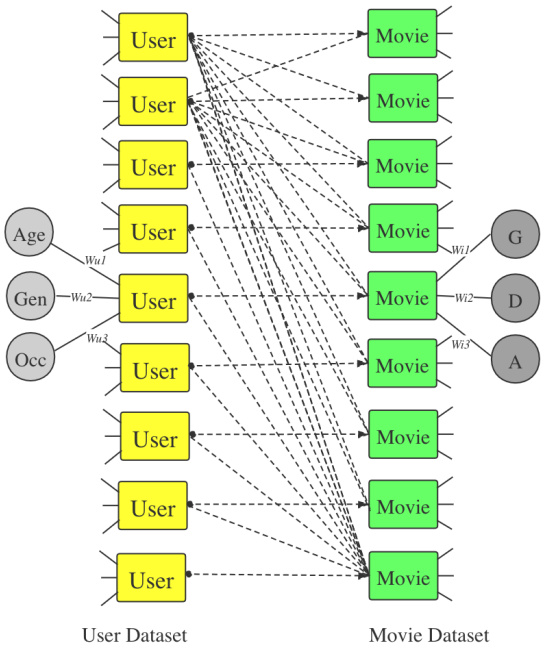
In terms of user trajectory, each time a user comes in, it is necessary to determine whether the user is a new one. Once a new user is found, a cold start strategy will be initiated. The system will guide the user to enter relevant personal information (gender, age, occupation). Then take a series of top-N movies from the existing movie data and let the user choose some movies he likes. After selecting n movies (in the subsequent prototype, we set n to 10 for easy description and testing). The system will use the previously obtained personal information of the user and the selected n favorite movies as input to build a user portrait, that is, the "Build user" part in the system architecture diagram. If the user is not involved in the cold start problem, the user goes to the "Build User" process directly. This part of the user data is the existing user data obtained from the user table in MovieLens. In summary, the existing user data in MovieLens and newly entered user data constitute the entire user database of the system together.

There are two types of labeling for movies, which are the characteristics of the movie itself (name, genre, director, actor) and the behavior characteristics of the movie in the system.

Movie characteristics:

The properties of the movie itself do not need to be updated frequently. These data will hardly change or need to be updated after the first input into the system. The genre of the movie and its release year can be retrieved from MovieLens.

Figure 3.2: UMNN Module



The movie data of MovieLens does not include the director, actor, writer, poster and other data of the movie. In order to obtain more useful movie metadata, so that the system we are constructing can have richer information and data to show, we use OMDb API which is a third-party RESTful web service to get movie information. The name and year of each movie are used as query inputs to obtain the director, actor, writer, and poster addresses of the movie. The new data corresponding to all movies in the entire movie table is re-integrated into a new movie table. The data items constituting the new movie table are: (iid, title, year, genre, director, actor, writer). The new movie table constitutes the "Movie Database" part in the system architecture

diagram.

Movie behavior characteristics:

Movie behavior characteristics refer to information such as the movie being clicked and rated in the system.

When the movie and user characteristics are collected, the "Movie Database" and the "User Database" are formed. These two components can be combined to build a model training set. All data in user database and movie database are structured as shown in the figure UMNN Module. The data structure in the rating table contains someone user's rating of someone movie. The data of the rating table is used as a mapping from the user dataset to the movie dataset. All user nodes and movie nodes that are related in the rating table are connected. All connections in the user dataset and movie dataset have weight values, and their initial value is set to the user's rating of the movie. In summary, the training set UMNN in the system architecture diagram is finally molded. With some machine learning algorithms provided by pytorch, by multiple rounds of training on UMNN, a UMNN with a series of new weights after fitting is constructed. The UMNN model is used as the core module of the recommendation system. When the user selects N movies he likes in the User Interface, the User Interface sends the selected movie id list as input to the UMNN, and the UMNN sends the recommended id list to the User Interface, and the User Interface finally shows the recommended movies' Information.

3.1.3 Frontend and Backend

Flask is a web micro-framework implemented by Python. The "micro" means that Flask aims to keep the code concise and easy to extend, allowing developers to quickly implement a website or web service using the Python language. It is currently a very popular web framework that uses the Python programming language to implement related web service. Based on Werkzeug and Jinja's lightweight WSGI web application framework, the main feature of the Flask framework is that the core structure is relatively simple, but has strong extensibility and compatibility. Programmers can use Python to quickly implement a website or web service.

Compared with other lightweight web frameworks, the Flask framework has good extensibility, has a flexible Jinja2 template engine, and improves the reuse rate of frontend code. This is irreplaceable by other web frameworks. In fact, the frontend and Backend development can be done in Flask with Python at one time. The template and route modules provided by Flask serve as the frontend and backend functions in the traditional network architecture. We use sqlite3 lightweight database to store some data that needs to be persisted. The entire frontend and backend and database architecture is very lightweight and concise, as shown in the figure.

In our entire system, the development language for machine learning algorithms and data processing operations is also Python, which is one of the reasons we chose it as the web development language and the Flask framework as the web framework. The entire system is developed by a unified language, which makes the whole development and later maintenance extremely convenient and clear.

The basic operating mode of Flask is to assign a view function to a URL in the program. Whenever a user accesses this URL, the system will execute the view function bound to the URL in the route module, obtain the return value of the function, and display it on the browser. Usually this return value is the parsed html code. its working process is shown in the figure.

The Web Server Gateway Interface (WSGI) has been used as a standard for Python web application development. WSGI is a specification for a common interface between a web server and a web application.

Werkzeug is a WSGI toolkit that implements requesting service, responding objects, and building some utility functions. It makes that possible to build web frameworks on top of it. The Flask framework embeds Werkzeug as one of its foundations.

Figure 3.3: frontend and backend

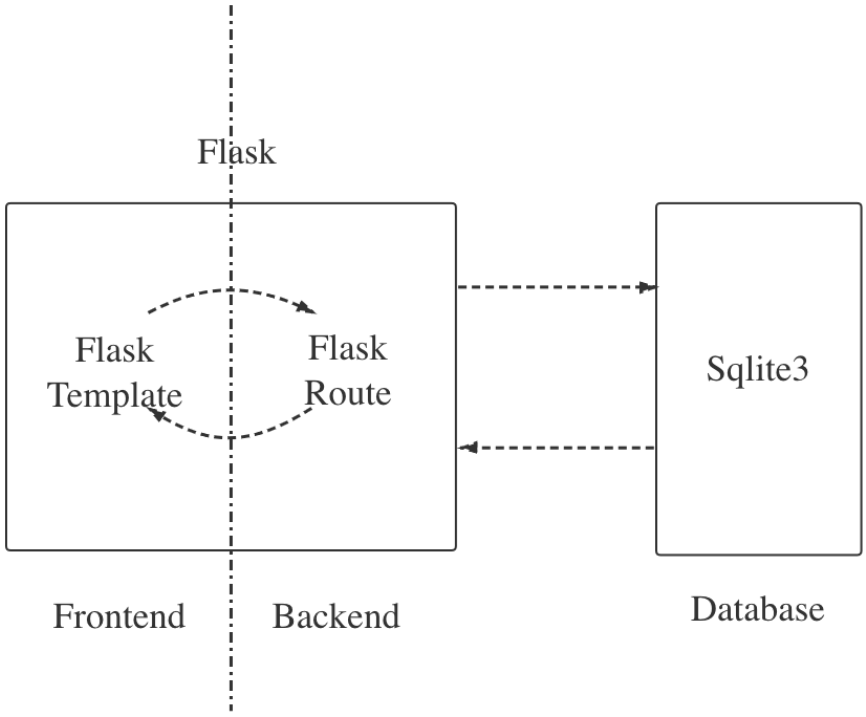
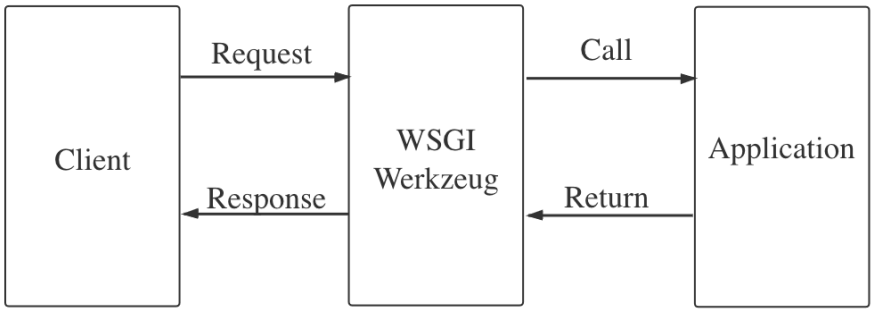


Figure 3.4: Flask working process



Jinja2 is a popular template engine for Python. The web template module combines templates with specific data sources to render dynamic web pages.

3.1.4 User Initialization

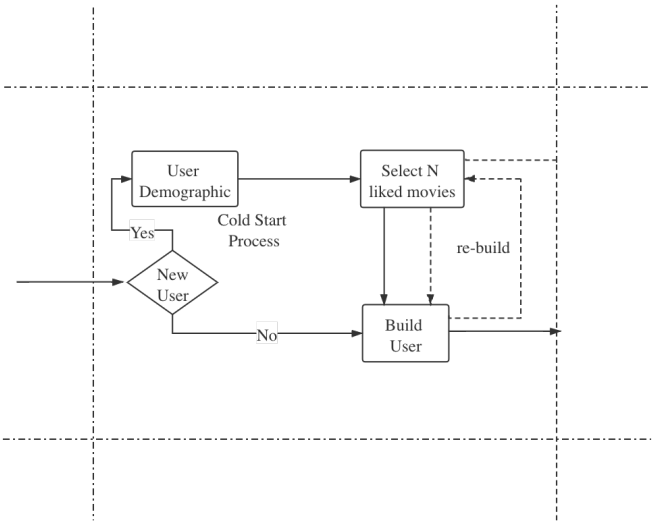
From a massive content library, the recommendation system recommends products that the user may like based on the user's current context information and past behavior. If a user's past behavior is empty, that is, the user's past behavior has not been recorded by the system, or he is simply a new user. A cold start problem occurs in this case.

The cold start problem of the movie recommendation system can be classified into new user cold start problem and new item cold start problem.

new user cold start problem:

lack of personal information data for new users, lack of records of people's interaction with movies, that is, visit clicks and ratings data.

Figure 3.5: Cold start process



new item cold start problem:

After the new movie is added to the system, due to the lack of visits, the recommendation will be inaccurate, even affects the chances of new items being recommended, and then continue to affect the access to new items, resulting in negative feedback. This will lead to a popularity bias problem. Movies that were originally popular are more likely to get most recommendations, and movies that have fewer new visits are less likely to be recommended.

Our system data comes from MovieLens. For the part of movie data, the movie data from MovieLens can be directly inputted into the system, so this paper does not involve the new item cold start problem. About the user data part, the user data in MovieLens can be used as the source of startup data. Only when a new user uses this system, a cold start strategy is needed. The specific steps have been described in the previous section. To summarize, the new user needs to enter relevant personal information (gender, age, occupation) and choose some movies he likes from a series of top-N movies. After the selecting procedure is completed, the previously obtained personal information and the selected n favorite movies are inputted to build a user portrait.

The method to solve the cold start problem is usually to obtain the demographic data of the new user, guide the user to interact with the system, record the user behavior, and then map the user behavior attributes to the attribute space. So that the new user can be associated with the existing user. That is, the new user node is associated with some movie nodes in the movie dataset, and at the same time, the new user node is indirectly associated with the existing user nodes in the UMNN by these movie nodes.

When solving the cold start problem, there are usually two problems to be aware of.

The first problem is that the data generated by user behavior, in cold start process, has sparse attributes. In large-scale application systems, users need to interact with a large number of items. For a specific user, there are only few interactions in the system, and the interaction with the item may be even less. So the user-item matrix will be very sparse. To deal with the sparseness of matrices, we can construct more efficient data structures, compress sparse rows and columns, or reduce dimensions through methods such as PCA and SVD.

The second problem is efficiency. The recommendation system is an online system. It needs the ability of immediateness and instant feedback. The user does not want or accept long waits and too many useless interactions. We want to guide users through as few as possible but sufficient operations to obtain as much and as effective user behavior data as possible. It will not cause users

to have a bad interactive experience because of tedious and redundant operations, but also can quickly and accurately construct user behavior portraits.

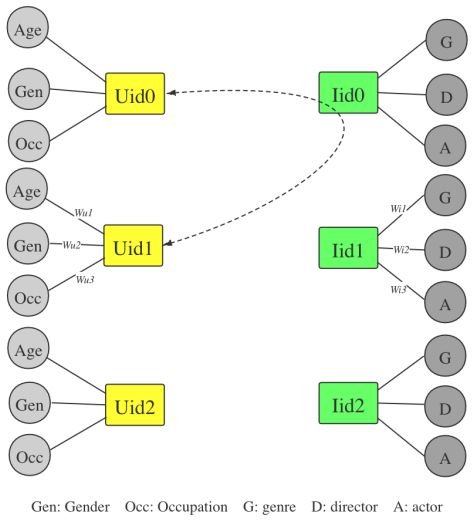
3.2 Recommendation Strategy

3.2.1 Recommendation Style

(a) **Popularity-Based**
There are two reasons for recommending based on popularity. First, popularity often represents the important characteristics of a product. Some users utilize extensive sources of information before making decisions to choose a movie, however, the others depend on simple and limited sources of information. But they all will be affected by the popularity of movies to some extent. Second, the popularity of a movie greatly influences user's decisions. From a psychological perspective, when recommending popular movies to users, even if they are not the type the user likes, users usually subconsciously give these movies a higher rating [Ahn, 2006]. So popularity-based recommendation is a very important and simple method in the early days. In this paper, popularity-based method is used as the basis for the other four recommendation methods.

(b) **User-Based**
The user-based recommendation strategy more considers the interests of new user and other

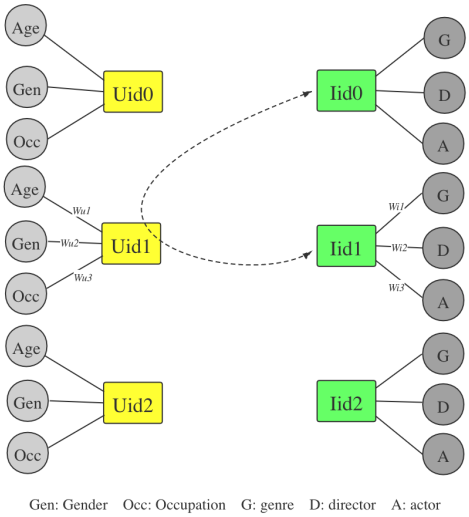
Figure 3.6: user-based recommendation strategy



users with the same hobbies, recommends items that other users like / visited to the new user, and has little to do with new user's current behavior.What will be recommended to a new user, depends on what the other users have visited before. The recommended items are the favorite items of users with the same hobby, so it has a hotspot effect. it can recommend the items that the other users have just visited. It has strong real-time performance, especially the newly introduced hot spots, which can spread quickly and solve the cold start problem of new-item.

(c) **Item-Based**
Item-based mainly uses users' historical interests to make recommendations. Recommending items that are similar to the user's history. This method has a lot to do with the user's current behavior. The similarity between the item recommended to the user and the item previously selected by the user is understandable by the user, which is called Interpretable.

Figure 3.7: item-based recommendation strategy



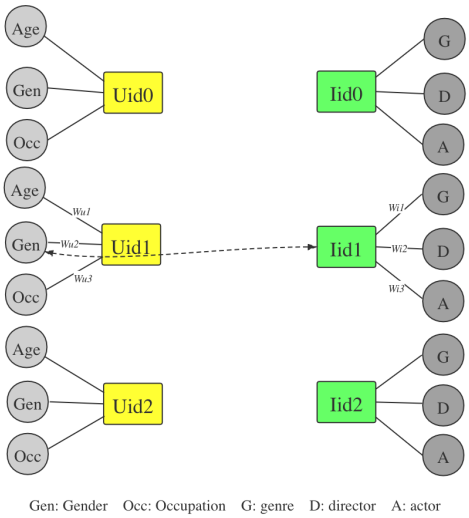
Most of the recommended items are not popular, but are related to the interests of users. This recommendation method has the highest accuracy when the user's interest is long-term and fixed. The significance of Item-based recommendation is to help users find items related to their interests. The recommended item is not related to user identity, so it is better to solve the problem of new users.

Badrul et al. [Sarwar et al., 2001] compared the performance of user-based and item-based and demonstrated that the item-based algorithm provides better quality of prediction than the user-based algorithm.

(d) **Demographic-Based**

According to the basic information of the system user, find out the relevance of the user,

Figure 3.8: demographic-based recommendation strategy

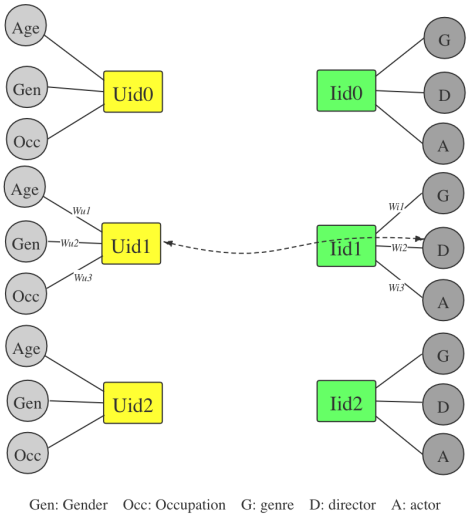


and then make recommendations. At present, it is rarely used alone in large systems, and it is usually used in combination with other recommendation algorithms. The usual method is to classify the user based on the user's registration information, and then recommend to the

user the items in the category to which he belongs. In this paper, we use gender, age and occupation of user as feature of demographic-based recommendation method.

(e) **Content-Based**
Based on the user's past browsing history, make recommendations to the user that he/she

Figure 3.9: content-based recommendation strategy



has not viewed. Recommendations are generally based on keywords or content features. For example, if a user has previously chosen to be interested in a certain director's movie, the user will be recommended with the other movie works of this director. The advantage of this method is that there is no popularity bias, items with rare features can be recommended, and user content characteristics can be used to provide recommendation explanations. The disadvantage is that the content must be machine-readable and meaningful, and the features of the recommended content need to be archived in advance.

Recommend movies that are similar to movies that users have liked. It is mainly based on the comparison of movie attribute information and user portrait information. The core problem is how to establish the association between movie attributes and user information. it assumes that users will rate items having alike features similarly [Safoury and Salah, 2013].

For various reasons, it is easier to collect the user's past behavior than to collect user information, but CF-based (user-based and item-based) has it's limitations. When the scoring is very sparse, the prediction accuracy will be greatly reduced. And the cold start of new products is also a problem for CF. In general, therefore, most of today's recommendation systems use a hybrid recommendation method.

3.2.2 Recommendation Explanation Strategy

In the construction of UMNN, all connection methods are recorded while the user node and the movie node are connected. When the User Interface sends a movie id list and UMNN sends a recommended id list, the corresponding connection method is transmitted to the User Interface as part of the response data.

We introduced the method for establishing a connection between the user node and the movie node in the Recommendation Style section. Each recommendation style can be relevant to a Recommendation Explanation Template.

The four recommended methods corresponding to the four connection methods in 4.1 are user-

UserNode–MovieNode–UserNode
MovieNode–UserNode–MovieNode
DemographicFeature–UserNode–MovieNode
UserNode–MovieNode–ContentFeature

Table 3.1: Four connection methods

based, item-based, demographic-based, and content-based. Establish the explanation templates for these four recommendation styles in table 3.2.

Recommendation Style	Explanation Template	Data Template	Example
user-based	(uid_1) is recommended with (iid_1) because (uid_2) is similar with (uid_1) and (uid_2) likes (iid_1).	Uid{uid-1}-Iid{iid1}-Uid{uid2}	You are recommended with "Resurrection Man (1998)" because user 5183 is similar with you and user 5183 likes this movie.
item-based	(iid_1) is recommended to (uid_1) because (iid_1) that is similar with (iid_2) which (uid_1) liked before.	Iid{iid1}-Uid{uid-1}-Iid{iid2}	"Woo (1998)" is recommended to you because that is similar with "Ice Storm (1997)" which you liked before.
demographic-based	(iid_1) is recommended to (uid_1) because (uid_1)'s (DemographicFeature-Type) is (DemographicFeatureValue).	Iid{iid1}-Uid{uid-1}-DFType{type}-DFValue{value}	"12 Monkeys (1995)" is recommended to you because your occupation is academic/educator.
content-based	content-based: (uid_1) is recommend with (iid_1) because (iid_1)'s (Content-FeatureType) is (Content-FeatureValue)	Uid{uid-1}-Iid{iid}-CFType{type}-CFValue{value}	You are recommended with "Resurrection Man (1998)" because the genre of the movie is Crime.

Table 3.2: Explanation Template

3.2.3 Recommendation Explanation Adaptation Strategy

The quality of a recommendation system usually requires an evaluation of the entire recommendation behavior, which runs through the user's whole interaction process. A recommendation system is good or not good, which is difficult to define. The algorithm is the core of the recommendation system, while a variety of other technologies and factors have a significant impact on personalized recommendations. The excellent recommendation system is a hybrid product of algorithms, various technics, and interaction designs.

From the use of search engines when there is less information at the beginning of the Internet, to the use of recommendation systems after more information appears on the Internet, to the applied of interpretable recommendation systems that provide a better interactive experience, the way users obtain information is constantly improved and optimized. In this paper, we use the adaptation method, which was rarely studied in previous related papers, to further increase the in-

teraction between the recommended content and the user, so that the user is more actively involved in the operation of the recommendation system, instead of as a simple message receiver. By allowing users to use our recommendation system multiple times, we collect the input of user behavior characteristics for each round as positive feedback to promote various aspects of the performance of the recommendation system.

We find some available and reasonable recommendation explanation adaptation strategies: adaptation for the way of showing, adaptation by trying to insert minority items, adaptation based on several rounds of user feedback.

Adaptation for the way of showing:

Adapt the shown of the explanation. The original movie recommendation explanation of the system is an understandable sentence using natural language, but sometimes the user is not interested in the uniform text description, and it is not necessary for the user to receive all the text information in the recommendation explanation. From linguistics, we know that the user only needs to read a few keywords, In most cases, he can connect, arrange, extend and imagine the keywords to understand the meaning of recommendation explanation.

We propose a possible solution for the adaptation view: use word cloud. We can combine the movie?s explanation content with the movie ?s metadata on wikipedia and OMDb. After doing some data processing, a word cloud can be generated. This word cloud could contain many important movie related tags or features. user can understand the reason for recommendation by the word cloud.

As shown in figure, there are two word clouds for the movie "Avengers Endgame" and the movie "The Dark Knight". we can respectively get the keywords from two word clouds and know the explanation of recommendation.

Figure 3.10: Word cloud

Adaptation based on several rounds of user feedback:

The core idea of this method is to record the relevant data of each round when the user interacts with the recommendation system for multiple rounds. The previous round of interaction data is used as input data to train and modify the user's portrait of the user stored in the system.

Taking the ratio of different recommended styles as an example, we can establish an adaptation

rule:

Algorithm 2: Adaptation Rule Algorithm

Result: propotion

$$\bar{score}(Sum) = \frac{\sum_{i=0}^{n} rating_i}{n}$$

$$\bar{score}(IUI) = \frac{\sum_{i \in IUI} rating_i}{|IUI|}$$

$$\bar{score}(UIU) = \frac{\sum_{i \in UIU} rating_i}{|UIU|}$$

$$\bar{score}(IUDD) = \frac{\sum_{i \in IUDD} rating_i}{|IUDD|}$$

$$\bar{score}(UICC) = \frac{\sum_{i \in UICC} rating_i}{|UICC|}$$

while

$$count_{sum} == 10$$

do

$$count_{IUI} \mathrel{+}= func_{propotion}(\bar{score}(IUI) - \bar{score}(sum))$$

$$count_{UIU} \mathrel{+}= func_{propotion}(\bar{score}(UIU) - \bar{score}(sum))$$

$$count_{IUDD} \mathrel{+}= func_{propotion}(\bar{score}(IUDD) - \bar{score}(sum))$$

$$count_{UICC} \mathrel{+}= func_{propotion}(\bar{score}(UICC) - \bar{score}(sum))$$

if

$$count_x < 0$$

then

$$count_x = 0$$

end

end

3.3 Machine Learning Algorithm

3.3.1 Tool and Library

PyTorch [Ketkar, 2017] and TensorFlow [Abadi et al., 2016] are currently the most popular methods for investigating deep learning and neural networks. PyTorch is more useful for researchers, enthusiasts and Individual developers to quickly build prototype in small-scale projects. TensorFlow is more suitable for large-scale deployments, especially when cross-platform and embedded deployments are required. In our research, PyTorch was selected for its vast repository of libraries to handle dataset preprocessing, statistical analysis, plotting, and more. [Paszke et al., 2019].

3.3.2 Algorithm

Algorithm 3: Path Generated Algorithm

Result:

$$P_{v_1}$$

$$\hat{r}$$

$$P_{v_1} = \begin{bmatrix} genre & actor & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ I_1 & I_2 & I_x & \dots & \dots \\ V_1 & V_1 & V_1 & V_1 & \dots \end{bmatrix} \quad L_P \times N_P$$

$$L_P : Length\ of\ Path$$

$$N_P : Count\ of\ Path$$

$$\begin{cases} p_1 = f_{path-encoder}(V_{genres}, V_{I_1}, V_1) \\ \dots \\ \dots \\ p_{N_P} = f_{path-encoder}(V_{director}, V_{I_N}, V_1) \end{cases}$$

$$att_{p_1} = \frac{e^{P_1 \cdot A^T}}{e^{\sigma(p_1 \cdot A^T)} + e^{\sigma(p_2 \cdot A^T)} + \dots + e^{\sigma(p_{N_P} \cdot A^T)}}$$

$$\mathbf{h}v_1 = \sum_{i=1}^{N_P} att_i \cdot P_i$$

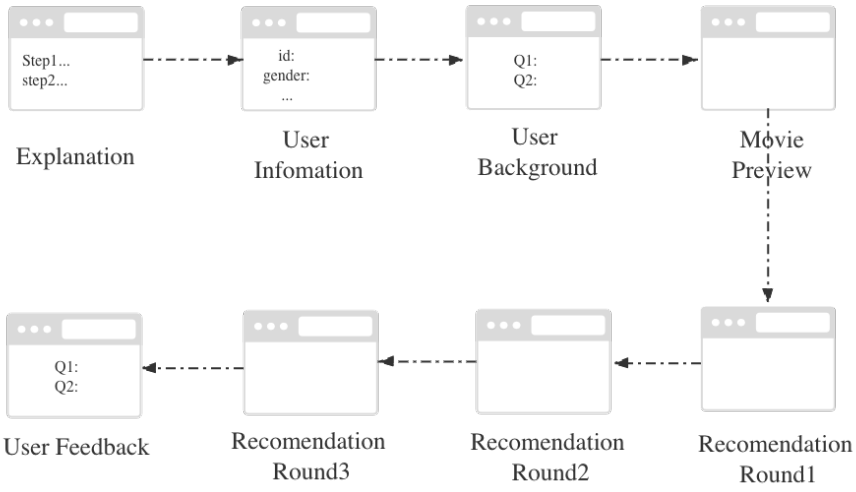
$$\hat{r} = \mathbf{h}_{user} \cdot \mathbf{h}_{att}^T$$

Training Weights:
A^T in algorithm 3 is a matrix that can be trained with large amounts of data.
σ is sigmoid function.
The att_{p_1} is also called a softmax(P,a) [Wu et al., 2016].

3.4 User Interface Prototype

UI flow:
In general, the UI flow is basically the same as the user data track in the system architecture diagram. Therefore, we can analyze the input and output data flow and its logical flow required by the prototype system according to the user data track, and design corresponding modules based on the functions required by the system.
Prototype:
The user interface prototype is designed as shown in figure 3.11.

Figure 3.11: User interface prototype



Page	Description
Explanation page	Some explanations about the User Study
User information page	For inputing personal data, like user id, gender, age and occupation.
User background page	The user will be asked about some background questions, like, how many does he know about Explainable Recommendation System?
Movie preview page	For building User Portraits, system will give user a series of movie posters and names. user can click to select the movies he likes from 6 options (selecting multiple, selecting none in 6 options and selecting all 6 options are allowed). Click the 'REFRESH' button to load the new 6 options. Repeat this selection process until the user has selected a total of 10 movies.
Recommendation round 1	The system will give user a series of recommendation movies and the explanations why these are recommended to the user. The user can click to select the movies he likes and give a score for the explanations.
Recommendation round 2	Same with Recommendation Round One, but the new recommendation movies are based on the user's choices and scores in Recommendation Round One.
Recommendation round 3	Same with Recommendation Round One and Two, but the new recommendation movies are based on the user's choices and scores in Recommendation Round Two.
User feedback page	The user's opinion about recommendation in the 3 test rounds.

Table 3.3: Prototype Description

4 Experiment

> That men do not learn very much from the lessons of history is the most important of all the lessons that history has to teach.
>
> Aldous Huxley

4.1 User Study Process

Let users use our recommendation system, record their feedback and answers to the questionnaires, and analyze data to evaluate the performance and interpretability of our recommendation system.

After reading the explanations about the user study, users will input some personal data (user id, gender, age and occupation) and then answer some background questions. For building user prtraits, system will give user a series of movie posters and names. user can click to select the movies he likes from 6 options. Click the REFRESH button to load the new 6 options and repeat this selection process until the user has selected a total of 10 movies. After that, the system will give user a series of recommendation movies and the explanations why these are recommended to the user. The user can click to select the movies he likes and give a score for the explanations. Repeat this recommendation step three times. Finally, the user will be required to answer some feedback questions.

4.2 likert Scale

The statistics of the questionnaire about system evaluation by using likert scale [Allen and Seaman, 2007] are shown in figure 4.1.

Figure 4.2 presents the results obtained from the preliminary analysis of points for 3 round recommendations from 20 users. We can see that the third round reported more high-points than the other two rounds. This shows that in our user study, the average score of users improved through three rounds of explanation adaptation.

4.3 User Study Metric

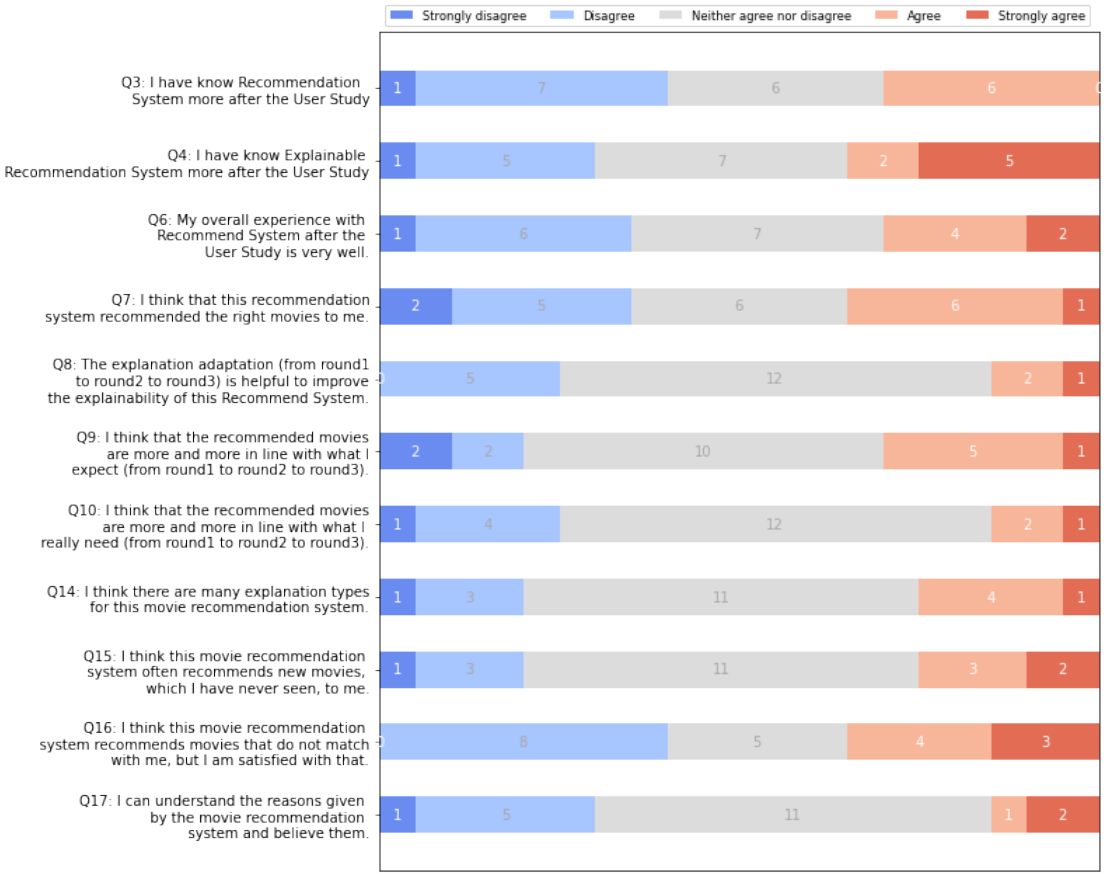
Correspondingly, the division of questions for evaluating the recommendation system is shown in the table 4.1.

Accuracy: Comparing the three questions about accuracy (Q7, Q9, Q10), we can find a very interesting phenomenon. Some users said that the recommendation system recommended the right movies to them, but they thought that the recommendation system did not recommend the movies they really expected and needed.

As Abraham argues in [Maslow, 1943] about Maslow's hierarchy of needs, from low to high, human needs are divided into physiological needs, safety needs, social needs, esteem and self-actualization needs.

Most recommendation systems, including our recommendation system, often recommend things that users may be interested in based on some data and records. This is certainly the right recommendation, but sometimes it may not be what users really expect and demand. The recommended things only satisfy the user's physiological needs, safety needs, and social needs. And higher-level esteem and self-actualization needs may become one of the possible development directions of future recommendation systems research area.

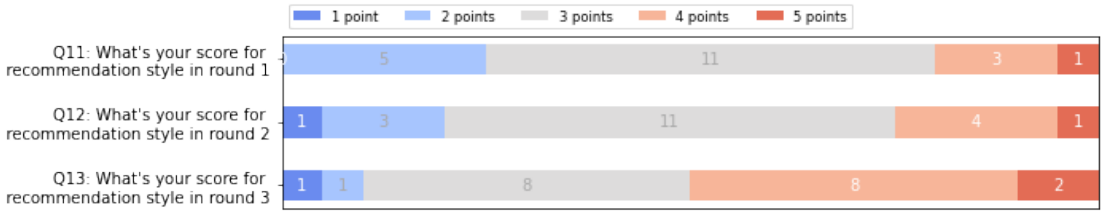
Figure 4.1: Feedback question about system evaluation



System Evaluation	Question	Result
Satisfaction	Q8	More than half of users choose a neutral attitude (12 in 20)
	Q7	7 users agree that the system recommend the right movies.
Accuracy	Q9	6 users agree that the system has recommended what they expect.
	Q10	3 users agree that the system has recommended what they really need.
Diversity	Q14	5 users agree
Novelty	Q15	5 users agree
Surprise	Q16	7 users agree
Trust	Q17	3 users agree

Table 4.1: Questions for system evaluation

Figure 4.2: Feedback question about 3 round recommendations



5 Evaluation

Figure 5.1: User behavior data

	user_id	movie_id	seen_status	explanation_type	explanation_score	user_study_round
1	1	126	1	UIU	3	1
2	1	307	0	UIU	2	1
3	1	950	0	UIU	2	1
4	1	1582	0	IUDD	4	1
5	1	1713	0	IUDD	5	1
6	1	1839	0	UIU	3	1
7	1	2113	0	UIU	3	1
8	1	2874	0	UIU	3	1
9	1	3006	0	UIU	3	1
10	1	3182	0	UIU	4	1
11	1	544	0	UIU	4	2
12	1	717	0	UIU	4	2
13	1	1385	0	UIU	3	2
14	1	1541	1	IUDD	4	2
15	1	2186	1	IUDD	4	2
16	1	2245	1	IUDD	3	2
17	1	2595	1	UIU	2	2
18	1	2744	1	UIU	5	2
19	1	3012	0	UIU	4	2
20	1	3157	0	UIU	4	2
21	1	23	0	UIU	3	3

Figure 5.1 is the structure of user behavior data. We collect:
the data of users' id
All the movies' id that selected by some users
Seen status (0:not seen, 1: seen)
Explanation type
Explanation score
Recommendation round number
We have already evaluated the data and have used it to build the Likert scale in the user study section.

6 Discussion

With the development of communication technology and data science, the relationship between people and information has evolved from one-way, people looking for information, to the current two-way relationship.

We compare the different ways in 5 recommendation styles in the recommendation style section. About the popularity-based recommendation, it often represents the important characteristics of a product. Users who depend on simple and limited sources of information, will be affected by the popularity of movies to some extent. The popularity of a movie will greatly influence user's decisions. Because of psychological factors, when recommending popular movies to users, even if they are not the type the user likes, users usually subconsciously give these movies a higher rating.

The user-based recommendation strategy more considers the interests of new user and other users with the same hobbies, recommends items that other users like / visited to the new user, and has little to do with new user's current behavior.What will be recommended to a new user, depends on what the other users have visited before.

Item-based mainly uses users' historical interests to make recommendations. Recommending items that are similar to the user's history. This method has a lot to do with the user's current behavior. The similarity between the item recommended to the user and the item previously selected by the user is understandable by the user, which is called Interpretable. The recommended item is not related to user identity, so it is better to solve the problem of new users.

According to the basic information of the system user, find out the relevance of the user, and then make recommendations. At present, it is rarely used alone in large systems, and it is usually used in combination with other recommendation algorithms. The usual method is to classify the user based on the user's registration information, and then recommend to the user the items in the category to which he belongs. In this paper, we use gender, age and occupation of user as feature of demographic-based recommendation method.

Based on the user's past browsing history, make recommendations to the user that he/she has not viewed. Recommendations are generally based on keywords or content features. The advantage of this method is that there is no popularity bias, items with rare features can be recommended, and user content characteristics can be used to provide recommendation explanations. The disadvantage is that the content must be machine-readable and meaningful, and the features of the recommended content need to be archived in advance.

Because various recommendation styles have particular advantages and disadvantages, in practical applications, a hybrid recommendation system is a better choice. The combination and adaptation with content recommendation and collaborative filtering recommendation is the common used method.

From a vertical perspective, over time and the development of computer technology. Human initiative is decreasing, and correspondingly, Internet initiative is increasing. This may be the trend of future recommendation system development.

With the development of Internet technology and computing science today, the speed of information generation and transmission is getting faster and faster, and the volume is getting larger and larger. The recommendation system plays a very important and irreplaceable role in the interaction between people and the Internet. But on the other hand, we can see that recommendation engines have great development potential to achieve the same magnitude as search engines.

7 Conclusion

> That men do not learn very much from the lessons of history is the most important of all the lessons that history has to teach.
>
> Aldous Huxley

We compared the different ways in 5 recommendation styles and proposed a hybrid recommendation system combined all the 5 recommendation styles. We provided an overview of the relationship among recommend system, recommendation explanation, and explanation adaptation. We explored the ways in which design an adaptation style for the recommendation explanation module and feedback scoring module. And we proposed a new methodology for the "adaptation rule" or "adaptation algorithm" of recommendation explanation.

We have understood the importance of the recommendation explanation for recommendation systems. Many current commercial recommendation systems rarely provide explanations, and even if a recommendation explanation is provided, most of them only adopt a single simple explanation mechanism. And these recommendation explanations are not fully utilized to improve the performance of the recommendation system.

In this thesis, we have experimentally confirmed that the research on the interpretability of the recommendation system helps the recommendation system to better interact with users, thereby generating accurate, effective and trustworthy recommendations.

The single recommendation method works poorly. Because various recommendation styles have particular advantages and disadvantages, in practical applications, a hybrid recommendation system is often used. We studied the combination and adaptation methods with content recommendation and collaborative filtering recommendation. The simplest way is to use a content-based method and a collaborative filtering recommendation method to generate a recommendation prediction result, and then use the explanation template to combine the results, and allow the interaction between explanation recommendation and user to adapt this result.

In the future, the research direction of explanation recommendation systems can be the following aspects:

(a) **Intelligence and visualization:**
We proposed a method " word cloud ", which is exactly a kind of visualization of explanation, that can be used for the adaptation of explanation.

The current recommendation explanations are more often presented with simple text explanations. For recommendation systems that store a large amount of information, the explanation content presented by the text is also bound to be huge. For example, a user browses a shopping website, the recommend explanation may only be 3 lines of text, but if hundreds of products, with recommending explanations, are displayed to the user at the same time. The amount of data the user will receive is very large, it will affect user-friendliness and greatly reduces the efficiency of recommendation explanation.

We can reduce the user's information receiving burden by visualizing the explanation. As recommendations become more precise and personalized, the content of recommendation explanations can be designed more intelligently. Trying to use deep learning methods to generate natural language that users can easily understand, to strengthen the interaction with the user and to improve user trust and satisfaction and effectively.

(b) **Design a more comprehensive interpretation template:**
For better interpreting and promoting the effectiveness of the recommendation system, in

addition to considering the evaluation criteria of the explanation and clearly explaining it, the presentation method of the recommendation system, interaction between users and explanations, and other aspects should also be taken into account.

(c) **More detailed classification:**
With the continuous development of recommendation systems, when new types of recommendation algorithms appear, new types of explanation also appear. To make the recommend explanation classification more general and to include a new type of interpretation, the classification dimension of interpretation will need to be further expanded, not just based on the three characteristics of users, products, and content. New dimensions such as time, place, and emotion may be added in the future.

(d) **Dive deeper into performance comparisons between explanation styles:**
At present, although there are evaluation standards for recommendation explanation, the research results in this area are still lacking for the performance comparison of different explanation types based on the role of different goals. And if we can accurately compare the advantages and disadvantages of different explanation types under different requirements, it will help system designers to more accurately choose the appropriate recommendation explanation styles for different recommendation systems.

8 Appendix

8.1 Content of enclosed CD

All codes are on the CD, frontend, backend, database, stored in the 'movie_RS' folder directory. The electronic version of the thesis is stored in the 'thesis' folder directory.

8.2 Kernel Code

```
1
2  def forward(self, x, path_index):
3          if self.training:
4              path_num = path_index.shape[1]
5              path_index = path_index[:, np.random.choice(range(path_num),
       int(path_num * (1 - self.path_dropout)))]
6          x = F.dropout(x, p=self.dropout, training=self.training)
7          x = F.elu(self.conv1(x, path_index)[0])
8          x = F.dropout(x, p=self.dropout, training=self.training)
9          x, att = self.conv2(x, path_index)
10         return x, att
```

Code 1: generate_path

```
1
2  def build_user(self, iids, demographic_info):
3          """
4          Build user profiles given the historical user interactions
5          :param iids: user selected item ids, list
6          :param demographic_info: (gender, occupation), tuple
7          :return:
8          """
9          self.base_iids = iids
10         self.demographic_info = demographic_info
11         # Build edges for new user
12         self.new_user_nid = self.model.node_emb.weight.shape[0]
13
14         new_user_gender_nid = self.data.e2nid[0]['gender'][demographic_info
       [0]]
15         new_user_occ_nid = self.data.e2nid[0]['occ'][int(demographic_info
       [1])]
16         i_nids = [self.data.e2nid[0]['iid'][iid] for iid in iids]
17         row = i_nids + [new_user_gender_nid, new_user_occ_nid]
18         col = [self.new_user_nid for i in range(len(iids) + 2)]
19         self.new_edge_index = torch.from_numpy(np.array([row, col])).long()
       .to(self.device_args['device'])
20
21         # Build path begins and ends with
22         new_path_np = utils.path.join(self.data.edge_index, self.
       new_edge_index)
23         self.new_path = torch.from_numpy(new_path_np).long().to(self.
       device_args['device'])
24
25         # Get new user embedding by applying message passing
26         self.new_user_emb = torch.nn.Embedding(1, self.model.node_emb.
       weight.shape[1], max_norm=1, norm_type=2.0)
27         new_node_emb = torch.cat((self.model.node_emb.weight, self.
       new_user_emb.weight), dim=0)
28         self.propagated_new_user_emb = self.model(new_node_emb, self.
       new_path)[0][-1, :]
```

Code 2: build_user

```
1
2  def get_recommendations(self,rs_proportion):
3
4          iids = self.get_top_n_popular_items(500).iid
5          iids = [iid for iid in iids if iid not in self.recommended]
6          rec_iids = [iid for iid in iids if iid not in self.base_iids]
7          rec_nids = [self.data.e2nid[0]['iid'][iid] for iid in rec_iids]
8
9          mask = np.isin(self.data.path_np[0][-1, :], rec_nids)
10         full_path_index = torch.from_numpy(self.data.path_np[0][:, mask]).
       to(self.device_args['device'])
11         propagated_node_emb = self.model(self.model.node_emb.weight,
       full_path_index)[0]
12         rec_item_emb = propagated_node_emb[rec_nids, :]
13         est_feedback = torch.sum(self.propagated_new_user_emb *
       rec_item_emb, dim=1).reshape(-1).cpu().detach().numpy()
14         rec_iid_idx = [i for i in np.argsort(est_feedback)]
15
16         rec_iids = [rec_iids[idx] for idx in rec_iid_idx]
17         exp_tuple = [self.get_explanation(iid) for iid in rec_iids]
18         exp, expl_types = [_[0] for _ in exp_tuple], [_[1] for _ in
       exp_tuple]
19
20         iui_rec_index = [idx for idx, expl_type in enumerate(expl_types) if
        expl_type == 'IUI'][:rs_proportion['IUI']]
21         iui_rec_iids = [rec_iids[idx] for idx in iui_rec_index]
22         iui_rec_exp = [exp[idx] for idx in iui_rec_index]
23
24         ...
25
26         temp_final_rec_iids = iui_rec_iids + uiu_rec_iids + iudd_rec_iids +
        uicc_rec_iids
27
28   ...
29
30         return rec_item_df, final_exp
```

Code 3: get_recommendation

```
1
2  def get_explanation(self, iid):
3
4          movie_nid = self.data.e2nid[0]['iid'][iid]
5          row = [movie_nid, self.new_user_nid]
6          col = [self.new_user_nid, movie_nid]
7          expl_edge_index = torch.from_numpy(np.array([row, col])).long().to(
       self.device_args['device'])
8          exist_edge_index = torch.cat((self.data.edge_index, self.
       new_edge_index), dim=1)
9          new_path_np = utils.path.join(exist_edge_index, expl_edge_index)
10         new_path = torch.from_numpy(new_path_np).long().to(self.device_args
       ['device'])
11         new_node_emb = torch.cat((self.model.node_emb.weight, self.
       new_user_emb.weight), dim=0)
12         att = self.model.forward(new_node_emb, new_path)[1]
13         opt_path = new_path[:, torch.argmax(att)].numpy()
14
15         e = self.data.nid2e[0][opt_path[0]]
16
17         if e[0] == 'uid':
18             expl = 'Uid0--Iid{}--Uid{}'.format(iid, e[1])
19             expl_type = 'UIU'
```

```
        elif e[0] == 'iid':
            expl = 'Iid{}--Uid0--Iid{}'.format(
                iid,
                e[1])
            expl_type = 'IUI'
        elif e[0] == 'gender' or e[0] == 'occ':
            expl = 'Iid{}--Uid0--DFType{}--DFValue{}'.format(
                iid,
                e[0],
                e[1]
            )
            expl_type = 'IUDD'
        else:
            expl = 'Uid0--Iid{}--CFType{}--CFValue{}'.format(
                iid,
                e[0],
                e[1]
            )
            expl_type = 'UICC'

        return expl, expl_type
```

Code 4: get_explanation

8.3 Tasks and Questions in User Study

8.3.1 Background Questions

1. I will watch one movie every

   O 1 day O 3 days O 7 days O 14 days O 30 days

2. I will visit a movie recommendation website I think every

   O 1 day O 3 days O 7 days O 14 days O 30 days

3. I know Recommendation System

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

4. I know Explainable Recommendation System.

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

5. I will use a Recommend System once every

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

6. How would you describe your overall experience with Recommend System?

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

7. I hope that this recommendation system will recommended the right movies to me.

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

8.3.2 Feedback Questions

1. I will watch one movie in the future every

   O 1 day O 3 days O 7 days O 14 days O 30 days

2. I will visit a movie recommendation website I think in the future every

   O 1 day O 3 days O 7 days O 14 days O 30 days

3. To what extent do you think that the proposed user segments can increase the requirement accuracy and correctness in this project?

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

4. I have know Recommendation System more after the User Study

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

5. I have know Explainable Recommendation System more after the User Study.

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

6. How would you describe your overall experience with Recommend System after the User Study.

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

7. I think that this recommendation system recommended the right movies to me.

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

8. The explanation adaptation (from round1 to round2 to round3) is helpful to improve the explainability of this Recommend System.

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

9. I think that the recommended movies are more and more in line with what I expect (from round1 to round2 to round3).

   O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

10. I think that the recommended movies are more and more in line with what I really need (from round1 to round2 to round3).

    O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

11. What's your score for recommendation style in round 1?

    O 1 O 2 O 3 O 4 O 5

12. What's your score for recommendation style in round 2?

    O 1 O 2 O 3 O 4 O 5

13. What's your score for recommendation style in round 3?

    O 1 O 2 O 3 O 4 O 5

14. I think there are many explanation types for this movie recommendation system.

    O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

15. I think this movie recommendation system often recommends new movies, which I have never seen, to me.

    O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

16. I think this movie recommendation system recommends movies that do not match with me, but I am satisfied with that.

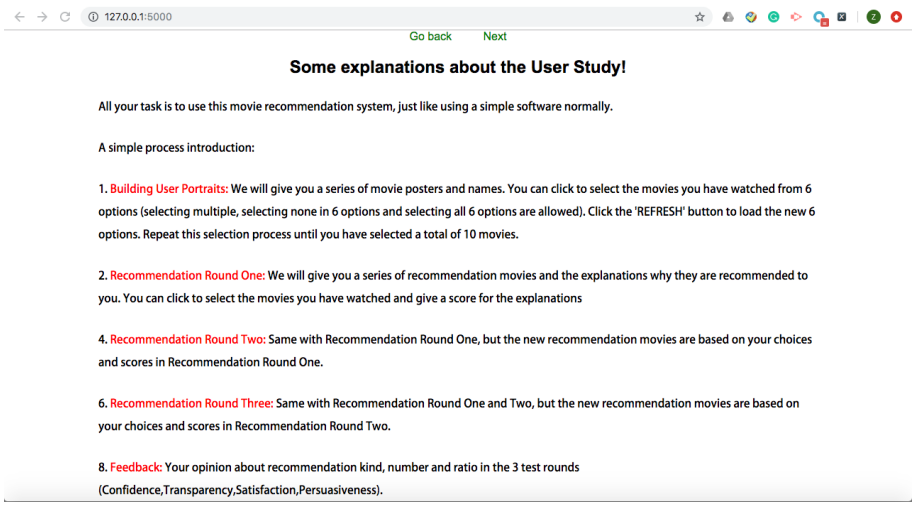
    O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

17. I can understand the reasons given by the movie recommendation system and believe them.

    O Strongly disagree O Disagree O Neither agree nor disagree O Agree O Strongly agree

8.4 Web application UI

Figure 8.1: Explanation Page



8.5 Raw Data Illustration

Figure 8.2: User Information Page

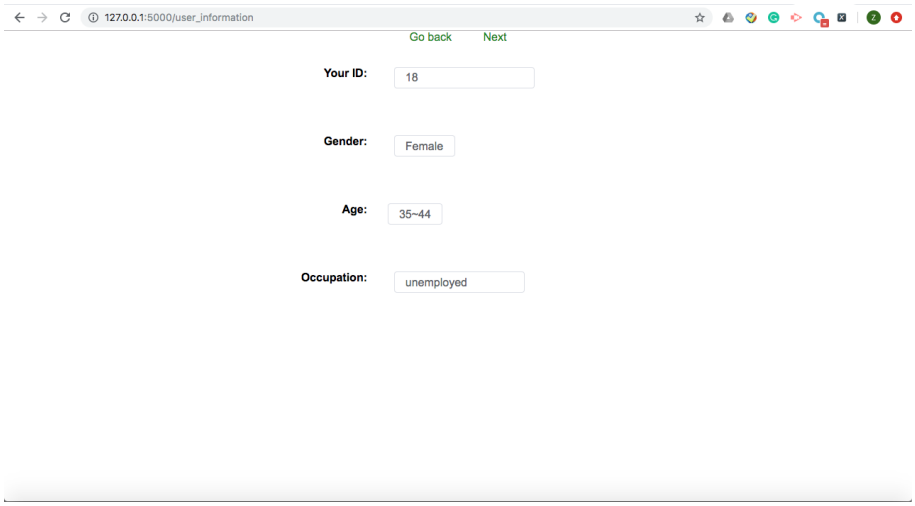


Figure 8.3: User Background Page

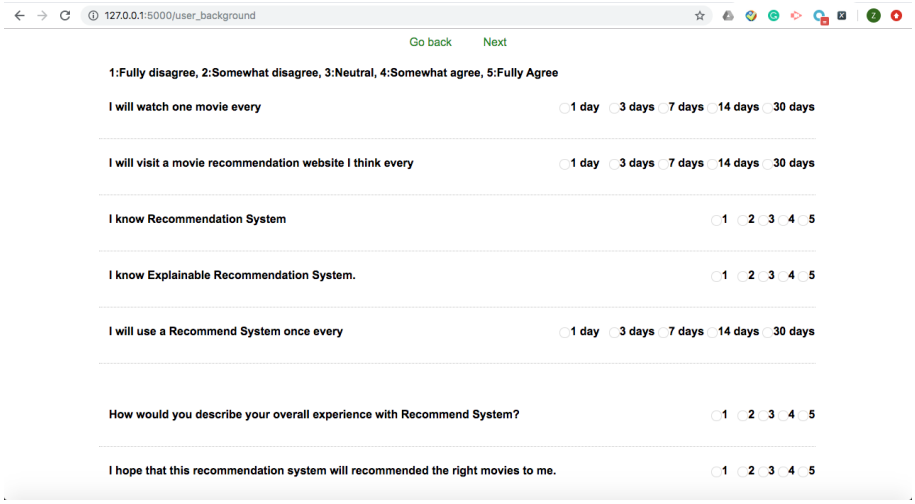


Figure 8.4: Movie Preview Page

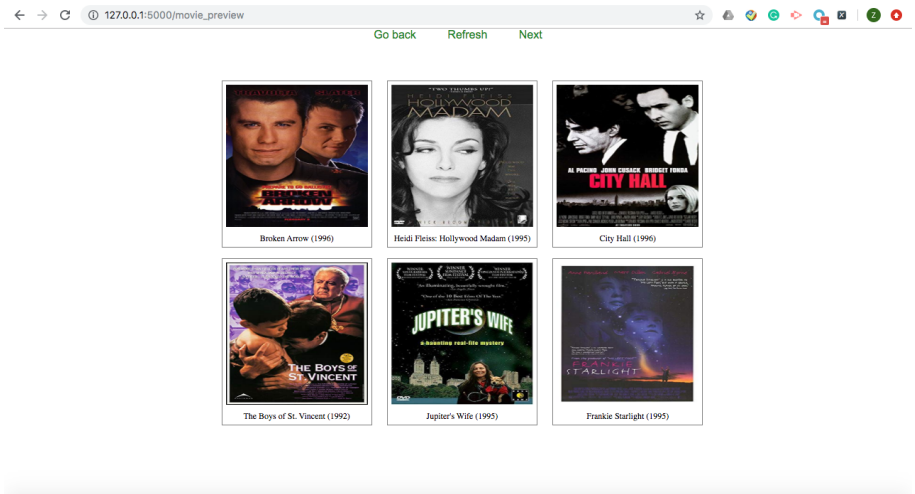


Figure 8.5: Movie Degree Page

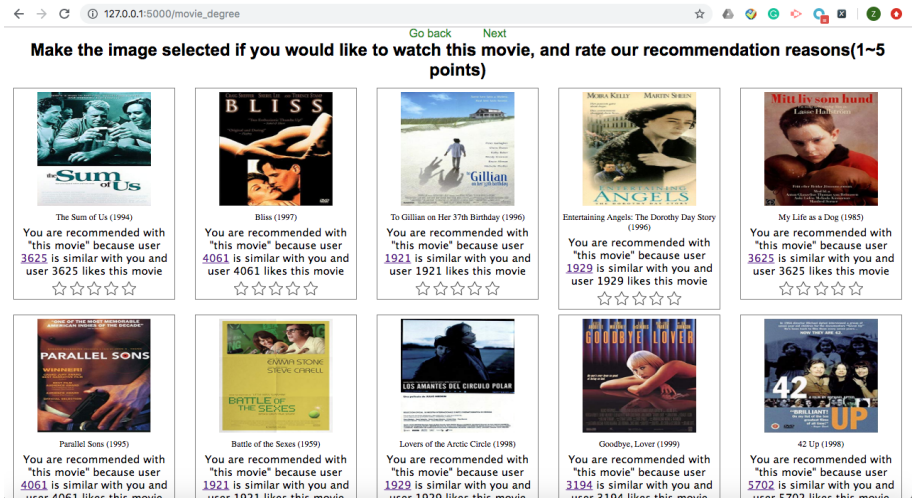


Figure 8.6: Movie Degree Page with scores

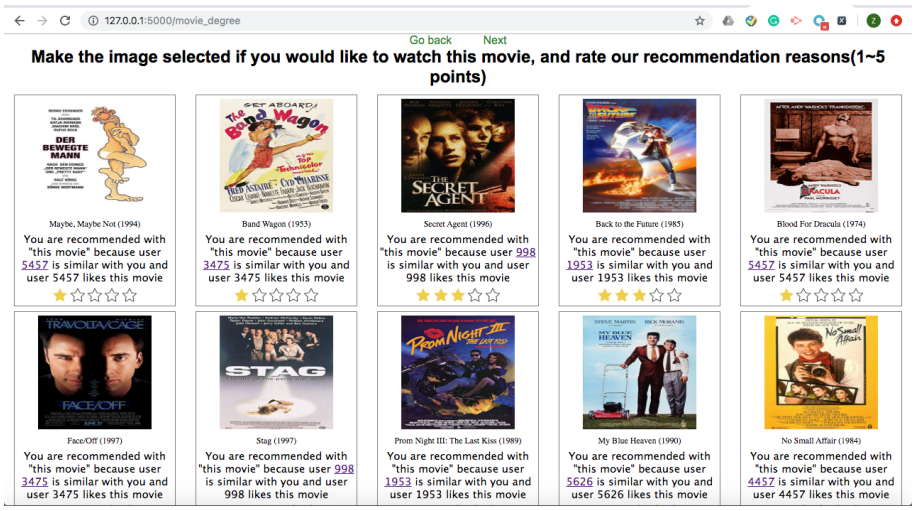


Figure 8.7: Coresponding User Inforamtion Page

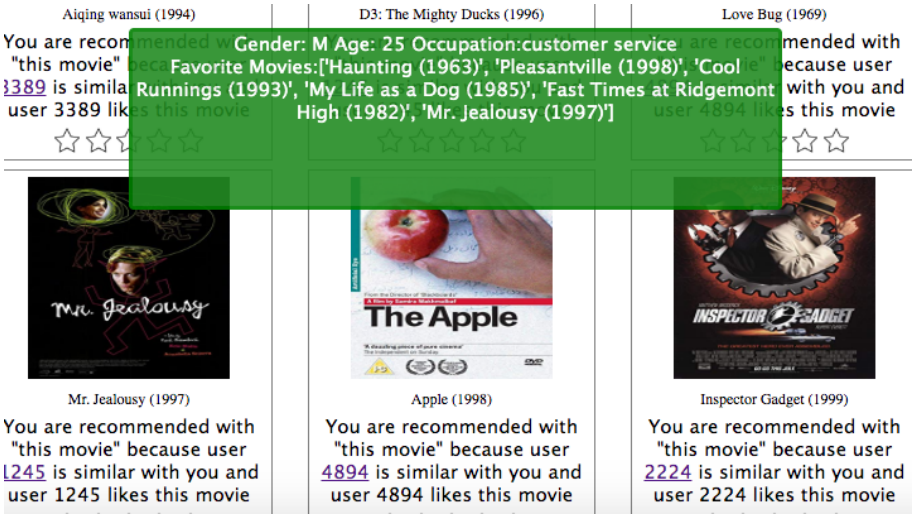
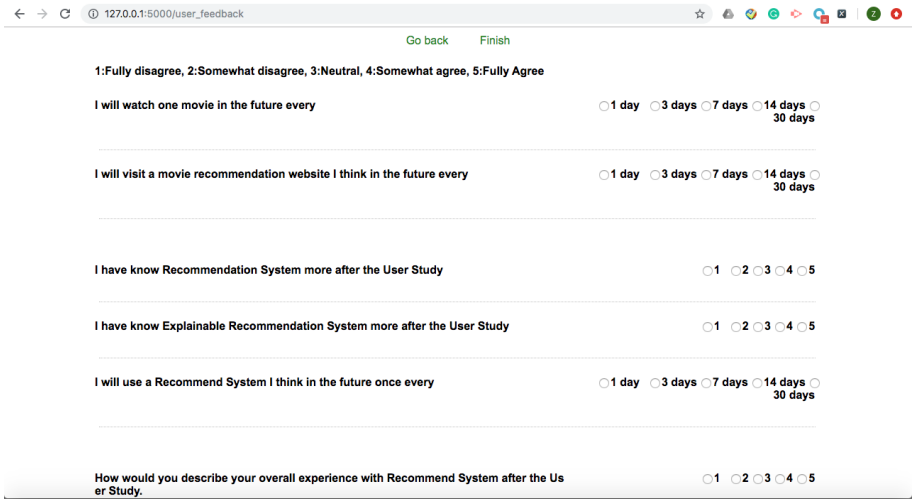


Figure 8.8: User Feedback Page



Bibliography

References

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283.

[Ahn, 2006] Ahn, H. J. (2006). Utilizing popularity characteristics for product recommendation. *International Journal of Electronic Commerce*, 11(2):59–80.

[Allen and Seaman, 2007] Allen, I. E. and Seaman, C. A. (2007). Likert scales and data analyses. *Quality progress*, 40(7):64–65.

[Anderson and Andersson, 2004] Anderson, C. and Andersson, M. P. (2004). Long tail.

[Balabanovic et al., 1996] Balabanovic, M., Shoham, Y., and Yun, Y. (1996). An adaptive agent for automated web browsing. Technical report, Stanford InfoLab.

[Cleger-Tamayo et al., 2012] Cleger-Tamayo, S., Fernández-Luna, J. M., and Huete, J. F. (2012). Top-n news recommendations in digital newspapers. *Knowledge-Based Systems*, 27:180–189.

[Duvenaud et al., 2015] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034.

[He et al., 2015] He, X., Chen, T., Kan, M.-Y., and Chen, X. (2015). Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670.

[Heckel et al., 2017] Heckel, R., Vlachos, M., Parnell, T., and Dünner, C. (2017). Scalable and interpretable product recommendations via overlapping co-clustering. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1033–1044. IEEE.

[Ketkar, 2017] Ketkar, N. (2017). Introduction to pytorch. In *Deep learning with python*, pages 195–208. Springer.

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[Maslow, 1943] Maslow, A. H. (1943). A theory of human motivation. *Psychological review*, 50(4):370.

[McAuley et al., 2015] McAuley, J., Targett, C., Shi, Q., and Van Den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52.

[Merton, 1968] Merton, R. K. (1968). The matthew effect in science: The reward and communication systems of science are considered. *Science*, 159(3810):56–63.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

[Ricci et al., 2011] Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer.

[Safoury and Salah, 2013] Safoury, L. and Salah, A. (2013). Exploiting user demographic attributes for solving cold-start problem in recommender system. *Lecture Notes on Software Engineering*, 1(3):303–307.

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

[Wu et al., 2016] Wu, Y., Li, J., Kong, Y., and Fu, Y. (2016). Deep convolutional neural network with independent softmax for large scale face recognition. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 1063–1067.

[Zhang and Chen, 2018] Zhang, Y. and Chen, X. (2018). Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*.