Final Year Project

ACME - Aspect Model Explainer for Top N Recommendation

Qihui Liu

Student ID: 16206519

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Professor Neil Hurley

UCD School of Computer Science

University College Dublin

May 21, 2020

Table of Contents

Abstract

Explainability has been identified as one of the key qualities of recommender systems. An explainable recommender will increase the user's trust in its recommendations. The explicit aspect model is a way to explain user's intention. Researchers at UCD have developed ACME - Aspect Model Explainer, which interprets the recommender system's decision via the explicit aspect model. The aim of this project is to develop a user interface for ACME, which will display the recommendations as well as the explanations made by ACME to the end-users. The project also needs to seek ways to allow users to change the recommendations through manipulating the explanations.

Chapter 1: Project Specification

Core

Develop a user interface that presents movie recommendations, along with an explanation and allows the recommendation to be changed through interaction with the explanation. The system should work for at least one underlying recommendation engine, and at least one explanation modality should be provided.

Advanced

An advanced system will be able to connect easily to any underlying recommendation engine. More than one explanation modality will be provided. A means of gathering evaluation data from end users will be provided.

Chapter 2: **Introduction**

A recommender system (RS), primarily used in commercial applications, is a system that seeks to predict the "rating" or "preference" a user would give to an item [2]. An RS needs to make personalized inferences about the user's current preferences based on the patterns of user interactions with the item catalogue in the past. Many AI techniques like deep learning are able to exploit these patterns, but these patterns are opaque to users. In other words, the user can only understand the input and the output of the RS algorithm, but it is difficult to understand the working principle inside the algorithm. This means many RS algorithms are actually black boxes. This has caused problems that end-users do not like to trust the decisions of RS algorithms, without understanding how they are made.

The purpose of XAI is to open up the black box of the RS algorithm, allowing end-users to better understand and more easily interpret their outputs. According to XAI research for RS, there are three explanation styles used for explaining recommendations : the User Style, which provides explanation based on similar users; the Item Style, which is based on choices made by users on similar items; and the Feature Style, which explains the recommendation based on item features. A Hybrid Style combines any of the above explanation styles [3].

My supervisor developed ACME - Aspect Model Explainer, a hybrid explanation style explainer. ACME largely provides a feature style explanation but also links the feature back to the user and to the recommended items. We will describe ACME in detail in the next Chapter. In general, ACME is used to form intuitive explanations for the recommendations made by a top-N algorithm. However, to understand how effective this explainer is, we need to design an appropriate user interface, which presents recommendations to users along with an aspect-based explanation also allows users to manipulate the explanation, and, in doing so, change the recommendation.

Our project will mainly focus on the case of movie recommendations. We rely on several baseline RS algorithms to firstly generate the top-N recommended movies. ACME then gives out the explanation of the recommendation by calculating some probabilities. The probabilities will be discussed in detail in the next chapter. Then the aim of this project is not only presenting the recommended movies but also seeking a way to visualize these probabilities and allowing the user to change them by using visual elements like charts, graphs, and other tools. By doing so, end-users can clearly understand how the recommendations are made, and also are able to change the recommendations via explanations given by ACME.

Chapter 3: Related Work and Ideas

In this chapter, we will first introduce the idea of reverse engineering, a general methodology used to interpret the decisions made by a black box recommender algorithm. Then we will elaborate on the explicit aspect model, a way to build up interpretations through people's inherent understanding of explicit aspects. After that, we will talk about ACME, an explainer based on the explicit aspect model. Finally, we will introduce a previous FYP, which has some connections to this work and provided a starting point. This project was about diverse recommendations and explain the similarities as well as the differences between that project and ours.

3.1 Reverse Engineering

Reverse Engineering is a common approach to understand a black box. It reconstructs an explanation for the output produced by a black box decision maker [1]. Such an approach can be classified as generalizable or not. A generalizable approach means it can be used to interpret any kind of black-box predictor[1].

One way to reconstruct the explanation is to build an interpretable model. For a possibly hard to interpret algorithm *f(.)*, build an interpretable model *g(.)* which tries to minimize the discrepancy between the result of *f(.)* and *g(.)*[4]. Less discrepancy means the output of *g(.)* is more faithful to that of *f(.)*, which means the explanation of the algorithm is more valid.

3.2 The Explicit Aspect Model

The Explicit Aspect Model is a kind of interpretable model. In the aspect model, each item is associated with a few explicit aspects that can be used to present the item type in an interpretable way that humans can make sense of. Such item features are also used to distinguish the user's interests, which drive the user's item selection process[5]. Therefore, in the explicit aspect model, the selection process can be expressed as two steps: the user first selects a particular aspect and then selects the item to satisfy that aspect, which leads to the following recommendation model.

$$p(rel_i|u) = \sum_{a \in A} p(a|u)p(rel_i|a)$$

Here, $p(a|u)$ is the probability aspect a is interesting to user u and $p(rel_i|a)$ is the probability item i is relevant to the aspect a. $p(rel_i|u)$, the probability item i is relevant to user u, is the sum of the product of the above two probabilities over all aspects.

Consider the case of movie recommendation. An aspect could be the genre of a movie. We assume that the movie "toy story" is of two genres: animation and comedy. $p(\text{toy story}|\text{animation})$ is the probability that "toy story" is relevant to animation, similarly, $p(\text{toy story}|\text{comedy})$ is the probability that "toy story" is relevant to comedy. For a particular user u, $p(\text{animation}|u)$ is the probability that u is interested in the animation genre and $p(\text{comedy}|u)$ is the probability that u is interested in the comedy genre. Suppose ACME has already calculated these probabilities: $p(\text{toy story}|\text{animation}) = 0.8$, $p(\text{toy story}|\text{comedy}) = 0.3$, $p(\text{animation}|u) = 0.2$, $p(\text{comedy}|u) = 0.4$. The probability that movie 'toy story' is relevant to user u could be calculated as follow:

$$
\begin{aligned}
& p(rel_{\text{toy story}}|u) \\
&= p(\text{animation}|u)p(\text{toy story}|\text{animation}) + p(\text{comedy}|u)p(\text{toy story}|\text{comedy}) \\
&= 0.8 * 0.2 + 0.3 * 0.4 \\
&= 0.28
\end{aligned}
$$

The above calculation shows how the recommendation are made through the user's taste to different aspects. Explanation given by the above recommendation process more understandable than other RS algorithm such as deep learning models. It is owing to the fact that the explanation lies in people's inherent understanding of genres such as 'comedy' and 'animation' and people have the ability to relate their tastes and interests to such genres. The goal of ACME then is to develop such an explicit aspect model, where recommendations are closely aliened to that of the RS that it explains.

3.3 The ACME system

ACME is an explainer based on the aspect model that can be used as an emulator of any top-N recommender algorithm that generates a score in order to rank its recommendations[6]. It can yield exactly the same ranking as the emulated algorithm. ACME can calculate the probabilities mentioned in the previous section and it is built on top of the recommender algorithm. The explanation for the recommendation is generated as follow: 1) compute $p(rel_i|u)$ from scores generated by the baseline recommender. 2) generate an aspect model factorisation to learn $p(a|u)$ and $p(rel_i|a)$. 3) Visualize and display the probabilities generated in the previous step with the recommendations to the end-user. 4) the end-user can change $p(a|u)$ and $p(rel_i|a)$ through the interface to regenerate the recommendations.

3.4 A project about diverse recommendations

Last year, a student's final year project was about developing a web-based application for recommendation with specified diversity. Our project is similar to that project except we are dealing with explainability rather than diversity. We studied the code implemented by the previous student and rebuilt her application.

Last year's project used a JAVA library called RankSys, which integrates several kinds of recommendation algorithms, to generate scores used for ranking[7]. The scores were generated offline and stored in a database. The application then connected to the database to give the recommendations(Figure 3.1). The application has an interface that allows the user to change the diversity using a slider. The recommendations will then be presented in the form of a table with attributes of movieID, title, and score. Our project is quite similar to that project, we will develop a web application and we will also use the RankSys library to generate the recommendations.

The biggest difference between last year's project and our project is that we are trying to build an explainable recommendation system. Therefore, what needs to be present to the end-user, will not only be the recommendation itself, but also the explanation of the recommendation. Apart from that, our interface also needs to allow end-users to change the recommendations by manipulating the explanations.

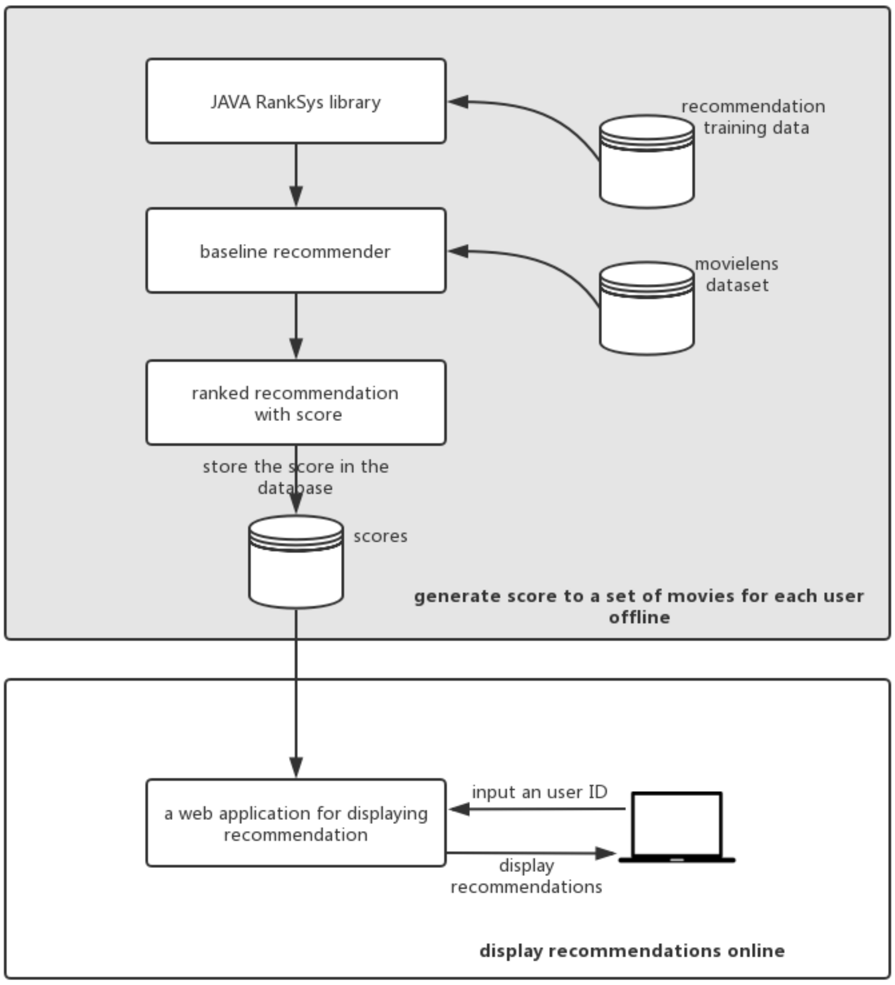

JAVA RankSys library
recommendation training data
baseline recommender
movielens dataset
ranked recommendation with score
store the score in the database
scores
generate score to a set of movies for each user offline
a web application for displaying recommendation
input an user ID
display recommendations
display recommendations online

Figure 3.1: Process flow of diverse recommendations system

Chapter 4: Data Context

As mentioned in section 3.3, ACME is an explainer that can be used as an emulator of any top-N recommendation that generates a score in order to rank its recommendations. The process for generating explanations can be divided into two steps(Figure 4.1): First, baseline recommender generates recommendations and recommendation scores from user preferences; Second, ACME generates explanations from recommendation scores. Specifically, user preferences are user's ratings and explanations refer to the two probability matrices $p(a|u)$ and $p(rel_i|a)$. Here we name $p(a|u)$ the "user aspect matrix" and $p(rel_i|a)$ the "item aspect matrix". As we have divided the whole process into two steps, we can divide the data into two parts, one is data for generating recommendations, the other is data for generating explanations.

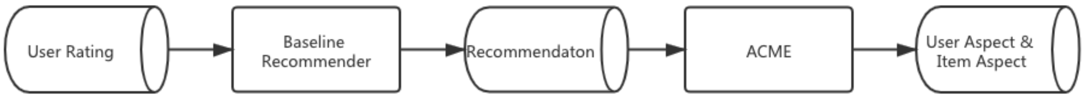


Figure 4.1: Basic flow of generating explanation

4.1 Data for Recommendation

We use MovieLens 100k data set, which can be downloaded from the website of grouplens[1]. The data set contains 100,000 ratings from 1000 users on 1700 movies. Each movie will be of one or multiple genres among all 19 genres. Ratings are made on a 5-star scale, whole-star ratings only. The data was collected through the MovieLens web site (movielens.umn.edu) during the seven-month period from September 19th, 1997 through April 22nd, 1998. This data has been cleaned up. Users who had less than 20 ratings or did not have complete demographic information were removed from this data set.

Among all 19 genres, there is one genre that indicates the genre of the movie is unknown. These movies need to be removed from the data set because explanations can only be generated for movies which are associated with at least one genre. The data set also includes the timestamp when the user rated a specific movie. This is not necessary for generating either recommendations or explanations so we will remove this field.

For more intuitive display of recommendations, we also want to display the poster of the recommended movie, in addition to name and recommendation score. Therefore, for each movie, we manually add another field called poster_url which is the url of the movie poster. The url is fetched by python scripts[12].

[1]https://grouplens.org/datasets/movielens/

Collection name	Attributes	Where the data comes from
ratingInfo	userID, movieID, rating	movielens
movieInfo	movieID, movieName, genres, posterUrl	movielens
recommendations	userID, movieID, score	Generated by baseline recommender
userAspects	userID, genreID, score	Generated by ACME
itemAspects	userID, genreID, itemID, score	Generated by ACME

Table 4.1

4.2 Data for Explanation

After the baseline recommender generates recommendations, ACME learns the user aspect probability $p(a|u)$ and the item aspect probability $p(rel_i|a)$. As mentioned in section 3.2, $p(a|u)$ is the probability that aspect a is interesting to user u and $p(rel_i|a)$ is the probability that item i is relevant to aspect a. User aspect matrix is a two dimensional matrix. The 1,3 element of user aspect matrix is the probability that aspect 3 is interesting to user 1. While item aspect matrix is a three dimensional matrix. The 1,3,4 element is the probability that item 4 is relevant to aspect 3 for user 1. To store the two probability matrices in a database, we will use compound keys. User aspect probability will have a userID and a genreID as its primary key; Item aspect probability will have a userID, a genreID and an itemID as its primiary key. Table 4.1 shows the detailed format of the data.

Chapter 5: Core Contribution

In order to build explanations of a recommender system model, we receive it as the input of these probability matrices and the goal is now to visualise these probability matrices in a way that is meaningful for to the users.

As this is a software engineering project, in this section, we will discuss the project from a set of software development activities, which are requirement analysis, design, implementation and evaluation.

5.1 Requirement Analysis

From the project specification, we can extract the following four requirements.

1. Develop a user interface that presents movie recommendations, along with an explanation, Allow the recommendation to be changed through interaction with the explanation.
2. The system should work for at least one underlying recommendation engine, and at least one explanation modality should be provided. An advanced system will provide more than one explanation modality.
3. An advanced system will be able to connect easily to any underlying recommendation engine.
4. An advanced system will provide a means of gathering evaluation data from end users.

However, the requirements from the project specification are not good enough. Good requirements should have the following characteristics: they should be atomic, unambiguous and testable[14]. Therefore, we need to modify the original requirement to meet these qualities.

Testable means testers should be able to verify whether the requirement is implemented correctly. To be testable, requirements should be clear, precise, and unambiguous. Some adjectives like robust, safe and effective can make a requirement untestable. Requirement 3 includes an adjective "easily" so it is not a testable requirement. It says that the system should connect easily to any recommendation engine. In other words, it means the system should also work for recommendation algorithm rather than algorithm that ACME uses. Therefore, we change the original requirement to 'the system should provide a way to add new model'.

Atomic means that the requirement should contain a single traceable element. Apparently requirement 1 and requirement 2 do not meet this quality. These two requirements include the words "and" and "along with", which indicates that each requirement contains more than 1 traceable element. Therefore we break the original requirement into several shorter sentences, each of which only contains 1 traceable element. See table 5.1 for details.

Interface	• present movie recommendations • present an explanation with the recommendations • provide more than one explanation modality • provide a way to interact with the explanation
Extensibility	• provide a way to add a new model
Evaluation	• provide a way to gather evaluation data from end users

Table 5.1: Requirements

5.2 Design

5.2.1 Desired Functionality

According to the requirements, we know that system needs to provide a means of evaluating the interface. If we want to evaluate the interface, we can consider using different ways to display the explanation. This also satisfies our requirement that the system should provide more than one explanation modality. The system is also required to provide a way to interact with the explanation so one explanation modality should be interactive.

Therefore, the modalities the interface would provide will be an interactive explanation and a non-interactive explanation. To examine whether the two modalities are effective or not, we also plan to have an interface which does not provide any explanation.

In our foundation report, we have already described a prototype in which users can use sliders to interact with explanations(Figure 5.1). This will be the example of the interactive explanation. As to the non-interactive explanation, we consider using text to show the explanation. For each recommended movie, an example textual explanation will be in the following: ***"The movie is recommended because it is an action and drama movie, and action is your top 1 preference, drama is your top 4 preference."***

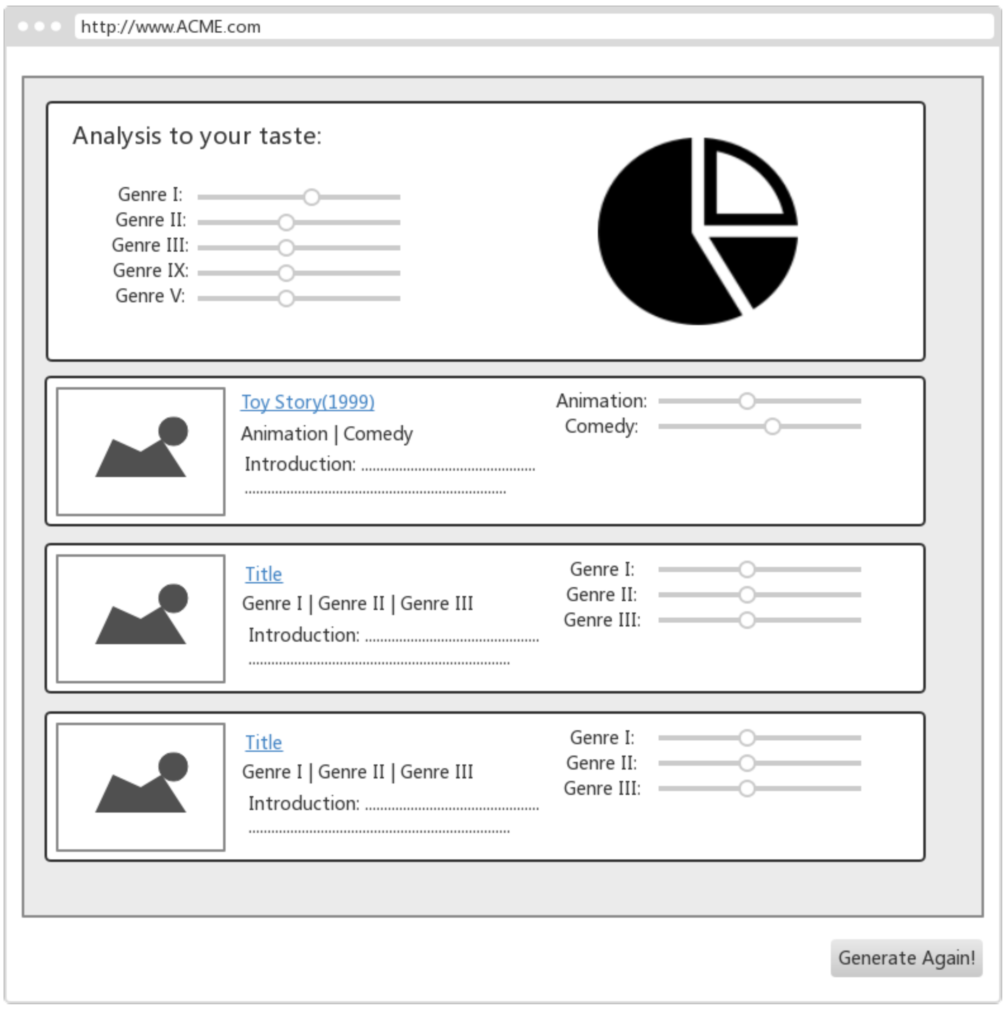


Figure 5.1: An example of using sliders to interact

According to the requirement, the system should also provide a way to add a new model. For adding a new explanation model, a straightforward solution would be firstly generating model data offline and then storing them in database. Using the File API in JavaScript, it's also possible for web content to ask the user to select local files and then read the contents of those files. The server side also has several ways to handle the uploaded file. Therefore, we could develop an interface for an administrator to upload model data to the database.

We already knew that ACME generates explanations according to user's previous preference to different genres so we can say that the genres of the recommended movies must be related to user's favoured genres in the past if the explanation model is effective. To compare them easily, we planned to display a table which shows the summary of user's preference after user enters the user ID. The summary will contain the number of movies in different genres and the average rating of movies in different genres. The detailed description of this function can be seen in table 5.2.

According to the requirement, we know that the system should provide a means of gathering evaluation data from end users. To meet this requirement, a survey page with an online questionnaire will be developed for gathering user feedback. Moreover, both questions and answers need to be stored in the database. By doing so, the administrator could update the questions and also gather answers from end users.

Function name	Description
Generate Recommendation	User can choose to generate recommendations without explanations or recommendations with textual explanation, or explanation with slider. For the last one, user could adjust probabilities by changing the sliders to update the recommendations.
Display movies rated by a user	User can view all movies rated by one user by entering the user id.
Summarise the rated movies	User can get a summary of rated movies in different genres. The summary will be in the format of a table, it will contain the number of movies in different genres and the average rating of movies in different genres.
Update model data	Administrator can upload a csv file which contains the ACME model data, such as user probability matrix and item probability matrix. After uploading the file, the file will be written into the database.
Provide feedback to the system	User can provide feedback to the system by filling in a questionnaire.
Gather feedback	Administrator can change the content of the questionnaire by updating the database. Administrator can also gather user's feedback by questionnaire.

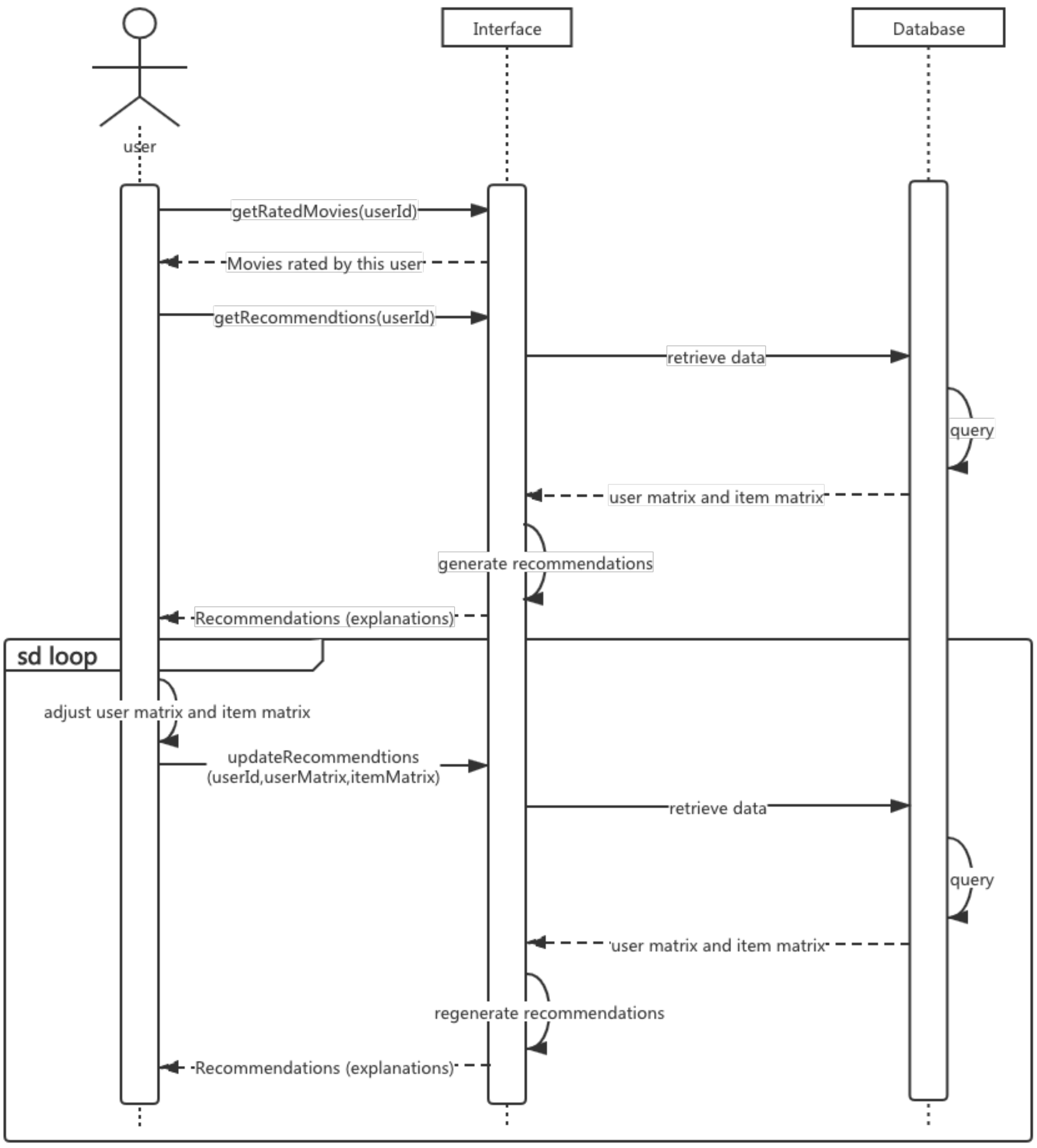
Table 5.2: System Functionality

5.2.2 Use Case Scenario

In this section, we will describe an example of how the user generates recommendations and adjusts the explanation via the interface. Among the three explanation modalities we designed previously, only interactive explanation allows interaction with the explanation.

Scenario: User gets recommendations with explanations and adjusts recommendations by modifying explanations.
1. The interface displays an input field for the user ID
2. The user enters 21
3. The interface displays all movies that user 21 has rated before, the interface also displays a table that contains the summary of the rated movies according to the genre.
4. A drop-down list with three options (no explanation, textual explanation, slider explanation) appears on the interface.
5. The user chooses the option 'slider explanation'.
6. The interface displays the recommendations and the explanations.
7. The user adjusts some sliders showing in the interface.
8. The user clicks the button 'Generate Again!'.
9. The interface updates the recommendations and the explanations.
10. The user can repeat step seven to step nine until they are satisfied with the explanation given.

This use case scenario can be described in the sequence diagram(Figure 5.2).



user
Interface
Database
getRatedMovies(userId)
Movies rated by this user
getRecommendtions(userId)
retrieve data
query
user matrix and item matrix
generate recommendations
Recommendations (explanations)
sd loop
adjust user matrix and item matrix
updateRecommendtions (userId,userMatrix,itemMatrix)
retrieve data
query
user matrix and item matrix
regenerate recommendations
Recommendations (explanations)

Figure 5.2: Sequence Diagram of getting recommendations and adjusting explanations

5.3 Implementation

5.3.1 Technology Stack

In this section, we describe the technology we used to develop this system and also illustrate the reason for choosing this technology.

Basically, the system consists of three parts: database, back-end and front-end. We discuss them separately in the following paragraphs. The summary can be found in table 5.3

The two most popular databases that are being used today are MongoDB and MySQL. The main difference between these two database is that MySQL is a relational database management system

Technology	Purpose	Justification
MongoDB	Document-oriented data store	Proven document-oriented NoSQL database. Suitable to store model data as documents and we can manage different explanation model as it is schema less.
Node.js	Server sider JavaScript Engine	Node.js enables fast and reliable software and the development is less time-consuming.
Mongoose	MongoDB-Node.js connector	mongoDB object modelling for node.js
Express	Node.js web framework	Express simplifies writing REST service in Node.js. It provides templating, error handling, providing lots of middle-ware.
Fast-csv	library for parsing and formatting CSVs	Suitable to parse newly-uploaded model data stored in csv format.
Bootstrap	front-end css framework	Lightweight front-end framework. Suitable to build responsive project on web.
D3.js	produce dynamic, interactive data visualizations in web browsers	Support multiple kinds of interactive data visualizations.

Table 5.3: Technology Stack

which stores data in tables and MongoDB is a NoSQL database that stores data as JSON-like documents[15]. Generally speaking, MySQL is more traditional and it is suitable for a legacy application. However, if you need a more flexible, schema-free solution that can work with unstructured data, MongoDB is better. Considering the data we used in this project, a schema-free solution is more suitable for us as we need to store data for different models. Moreover, considering the limited development time, MongoDB is better.

For the server-side technology, the most popular technology nowadays are Node.js and Spring Boot. Compared to Spring Boot, Node.js is more efficient and lightweight. As the system does not need a huge amount of computation, Node.js can fully meet the computation requirement of this system. What's more, npm, an online repository for the publishing of open-source Node.js projects, provides some very useful models such as Express and Mongoose, which could be used in our project. See table 5.3 for detailed descriptions of Express and Mongoose.

For the front-end technology, the most important thing we need to consider is how to visualise the data. A very powerful JavaScript library for data visualization is D3.js. D3.js supports multiple kinds of visualising, including sliders and pie chart shown in the interface prototype(figure 5.1).

Figure 5.3 represents the system we built. In general, the system consists of a front-end based on the Bootstrap framework, a back-end based on Node.js and a mongoDB database. In order to give the front-end app access to the database using HTTP request, Express is used to provide a REST API and Mongoose is used to connect MongoDB database and node.js. Besides, D3.js is used to produce interactive data visualisation.

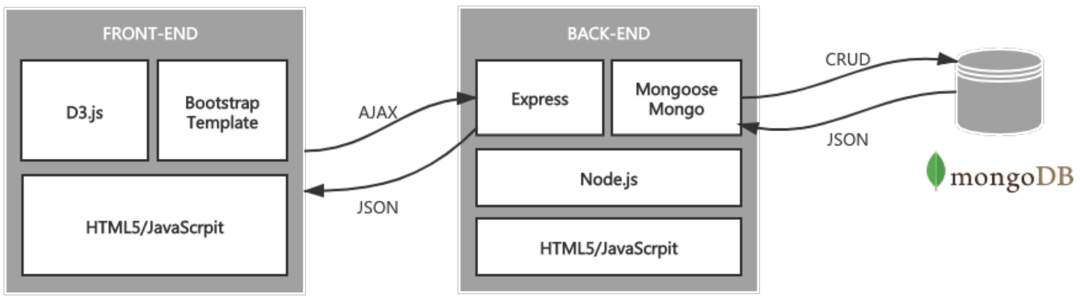


Figure 5.3: System Architecture

5.3.2 Database

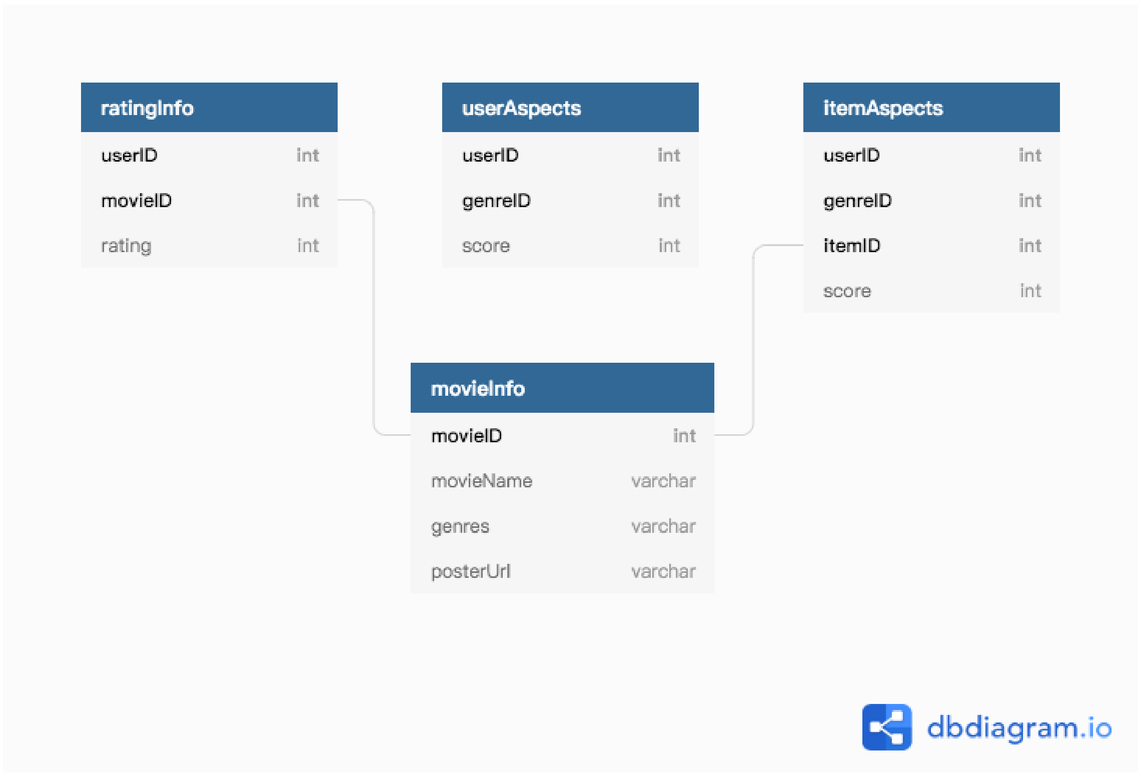


Figure 5.4: Database Architecture

In the chapter Data Context, we stated that data can be divided into five parts(Table 4.1), which are movies, user ratings, recommendations, user aspect probabilities and item aspect probabilities. Because ACME can factorize the relevance scores generated by the baseline recommender into the user aspect probability matrix and the item aspect probability matrix, ACME can also inversely get the relevance score from the two probability matrices. Therefore, we do not need to store the recommendations into the MongoDB database as recommendations can be generated only using the two probability matrices. As a result, there are four collections stored in MongoDB. Figure 5.4 shows the name, attributes and corresponding data types of the four collections.

Methods	Urls	Actions
POST	/users/display	Access rated movies
POST	/users/explanation	Access recommendations and explanations
GET	/users/survey	Access the questionnaire
POST	/users/submitsurvey	Submit the questionnaire
GET	/users/administration	Access administrator's content
POST	/users/fileupload	Upload model data

Table 5.4: APIs

5.3.3 API Design

Node.js is a JavaScript runtime environment that runs server-side. Within that environment, we can use JavaScript to build our software, our REST APIs, and invoke external services through their APIs. In this section, we define a list of REST APIs that the system needs to provide(Table 5.4).

5.3.4 Handle request using node.js and express

In this section, we give an example of how the system handles requests from the end-user. The main functionality of the system is to generate recommendations and explanations so we will talk about it in detail.

To generate recommendations and explanations, firstly the user needs to submit a form which contains the user ID and explanation type. By submitting the form, a HTML post request is sent to the url '/users/explanation'. On the server side, we also need a routing method to handle '/users/explanation'. The routing method specifies a callback function called when the application receives a request to the specified route and HTTP method. In other words, the application "listens" for requests that match the specified route and method, and when it detects a match, it calls the specified callback function. A route method is derived from one of the HTTP methods, and is attached to an instance of the express class. We can access the parameters of a post request through the req object.

```
router.post('/users/explanation',function (req,res,next){
    var type = req.body.explanationtype;
    var userID = req.body.userID;
    ...
});
```

Then we use Mongoose to retrieve data from the MongoDB database. We defined two callback functions retrieveUserAspect and retrieveItemAspect for retrieving user aspect probability and item aspect probability. Here find() is used to find a list of documents by its userID field. If the query is successful, the data object will be populated with the results of the query. We use a loop to iterate the documents and we can access the attribute of each document by dat.attributeName.

```
function retrieveUserAspect(userID, callback) {
    ua.find({ userID: userID }).exec(function (err, data) {
        ...
        let genres = {};
        for (let dat of data) {
            if (dat.score != 0) {
                genres[dat.genreID] = dat.score;
            }
        }
        callback(null, genres);
    });
};
```

Then we send the HTTP response to the client side.

```
res.send({
    recPro : rec.recPro,
    genrePro: rec.genrePro,
    genres: rec.genres,
    userID, userID,
    genreName,genreName,
});
```

Back on the front end, we use EJS to generate HTML markup with plain JavaScript. We used the <input> HTML elements to show the explanations. Below is the code for generating <input> elements.

```
<%for(var key in genres){ %>
    <div>
        <label><%= genreName.get(key) %></label>
        <input type='range' min='1' max='100' id='<%= "slider" + " " + key
            %>' value='<%= genres[key]'>
    </div>
<% } %>
```

Besides submitting forms, we also used Ajax for asynchronously updating web pages. Asynchronous means it is possible to update part of the page without reloading it. In our system, the recommendations and explanations are regenerated by asynchronous update.

```
$.ajax({
    url: '/users/update',
    type: 'POST',
    contentType: "application/json",
    data: JSON.stringify({
        genre_score: genres,
        genrePro: genrePro,
        userID: userID,
    }),
    success: function(){
    ...
    },
})
```

The JSON.stringfy() methods convert the JavaScript object or value to a JSON string. Then a HTML post request including the updated data is sent to the url '/users/update'. When the request succeeds, the function updates some HTML elements to show the updated recommendations and explanations.

5.3.5 Data Visualisation

In addition to using text to display information, the system supports also two kinds of graphic visualisation. one is a pie chart, the other is sliders. According to the interface prototype(Figure 5.1), the upper part of the page will be used to display the user aspect matrix and the remaining part will be used to display the item aspect matrix. The pie chart in the upper right corner will also be used to display the user aspect matrix. If the user adjusts the sliders of the user aspect matrix, the pie chart will change accordingly.

To implement the pie chart, we used the pie layout in D3.js.

```
var pie = d3.layout.pie()
```

To implement the sliders, we used the original HTML element <input type='range'>. JQuery's event methods can attach a function to an event handler for the input element. So when user adjusts the sliders, a function that is used to update the matrices will be called.

```
$ (".slider").change(function () {
    var index = this.id.replace("slider ", "");  \\get the index in the
        matrix
    var value = this.value;                      \\ get the updated value
    genres[String(index)] = Number(value);       \\update user aspect
        matrix
    change(randomData());                        \\update the pie chart
});
```

5.3.6 Technical Details

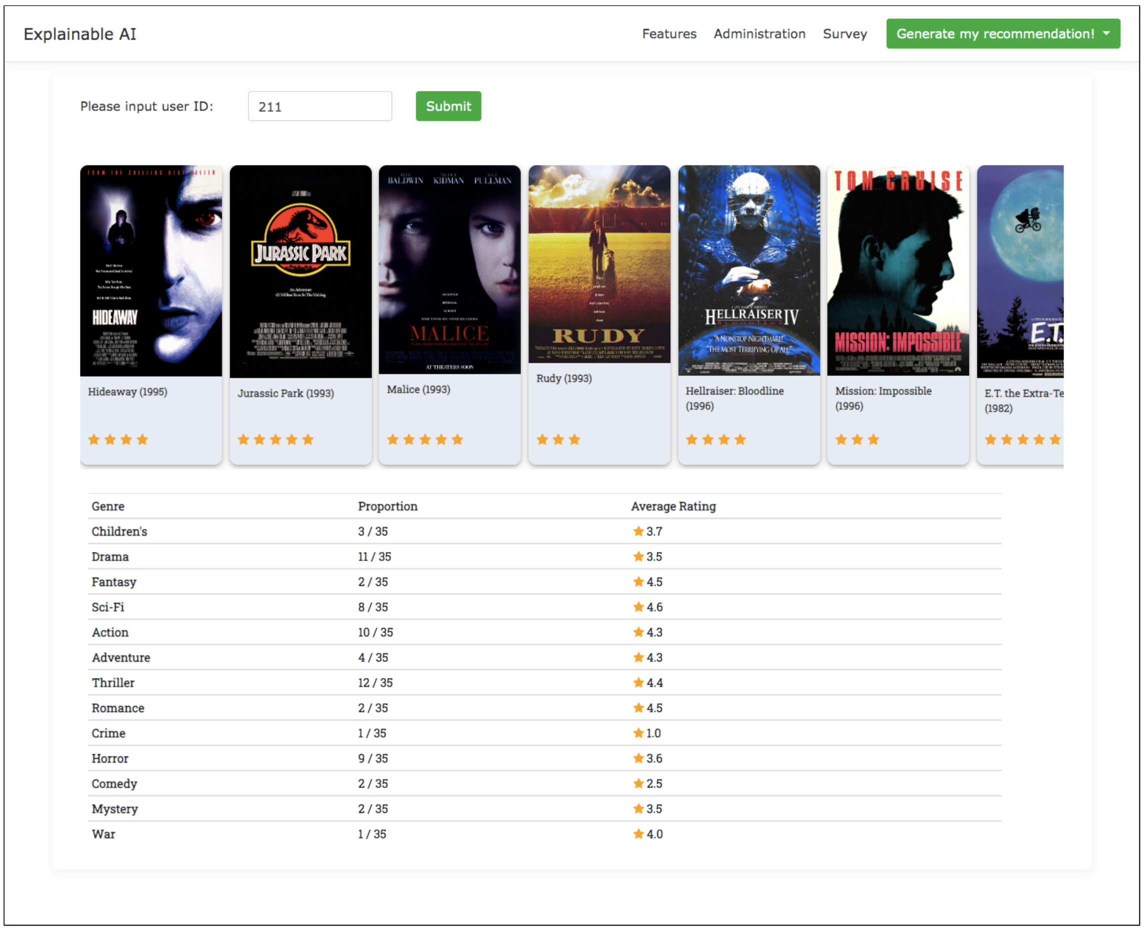
In this section, we will talk about some practical problems we encountered in the development.

According to the initial design, there are 18 genres in total and each user is interested in several of the 18 genres. It is possible that one user is interested in more than 10 genres. Under such case, the UI needs to display many sliders to show the probability $p(a|u)$. With so man sliders, the user may experience stress as too much information is available. This problem is known as information overload[13]. Therefore, in the actual implementation, if too may sliders need to be displayed, we only show the top 10 genres that the user is most interested in. As to item matrix, we take same measure to display sliders.

After the recommendations are generated for the first time, the user can update explanations by adjusting the sliders. For more convenient adjustments of slider, the system will adjust the matrix to a proper scale. For example, if $p(rel_i|a)$ equals 1, the probability will multiply by 0.8 to make sure the probability is always adjustable.

5.4 Example UI

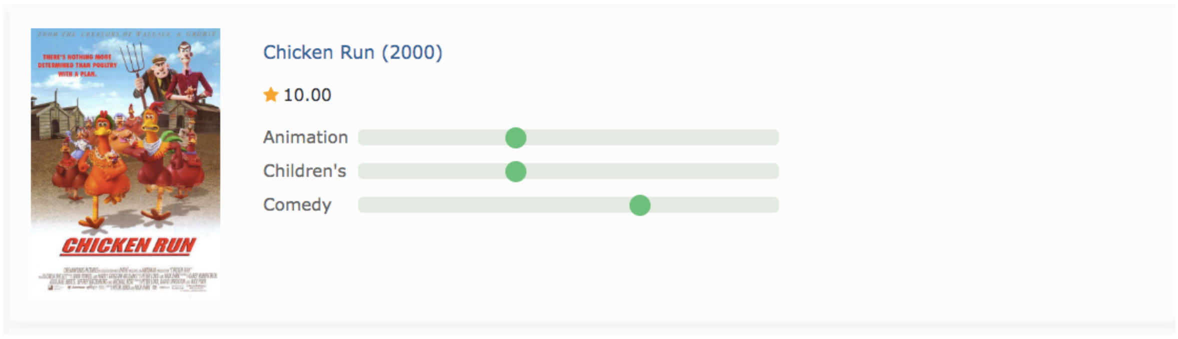
In this section, we will display some sample interface. The program for the user interface can be accessed from https://github.com/refrrir/FYP-explainableAI. To generate recommendations, the user firstly enters a user ID in the initial page then the interface displays all movies rated by the user, as well as a summary of user's preference(Figure 5.5). Then clicks the button 'Generate my recommendation' in the right up corner, a drop-down list with three options will appear. Choosing a different option will lead to different types of explanations(Figure 5.6).



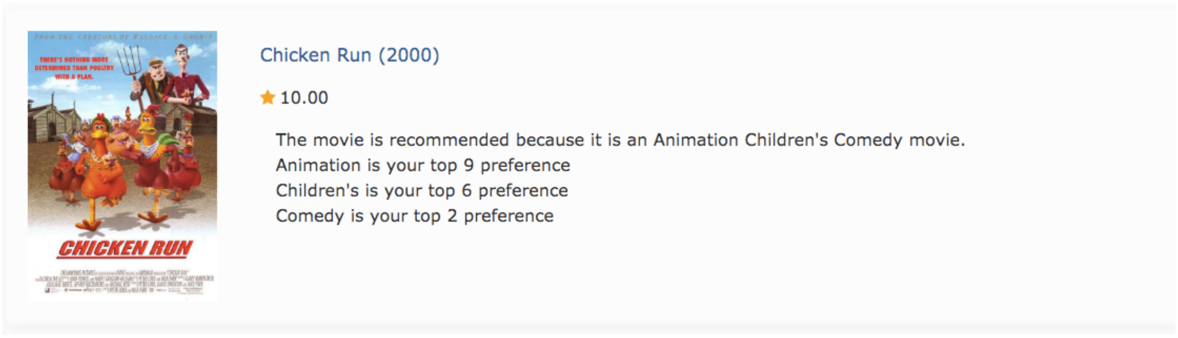
Hideaway (1995) ★★★★
Jurassic Park (1993) ★★★★★
Malice (1993) ★★★★★
Rudy (1993) ★★★
Hellraiser: Bloodline (1996) ★★★★
Mission: Impossible (1996) ★★★
E.T. the Extra-Te (1982) ★★★★★

Genre	Proportion	Average Rating
Children's	3 / 35	★3.7
Drama	11 / 35	★3.5
Fantasy	2 / 35	★4.5
Sci-Fi	8 / 35	★4.6
Action	10 / 35	★4.3
Adventure	4 / 35	★4.3
Thriller	12 / 35	★4.4
Romance	2 / 35	★4.5
Crime	1 / 35	★1.0
Horror	9 / 35	★3.6
Comedy	2 / 35	★2.5
Mystery	2 / 35	★3.5
War	1 / 35	★4.0

Figure 5.5: Movies rated by user with ID 211



(a) slider explanation

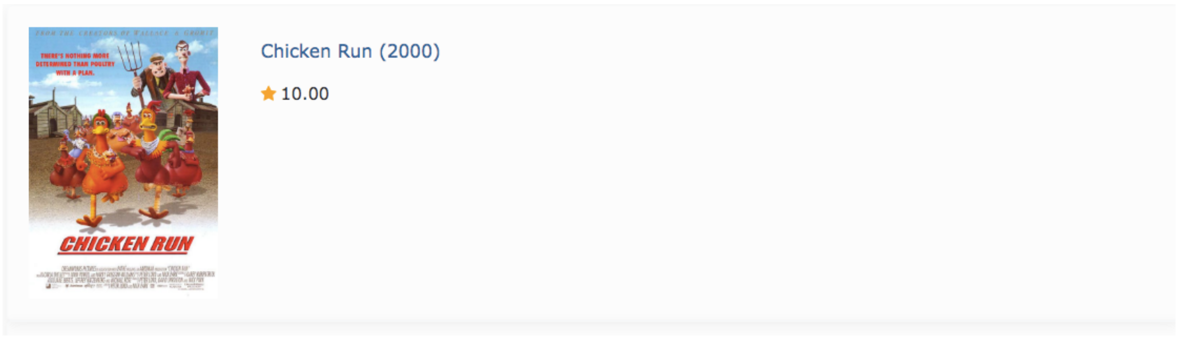


Chicken Run (2000)

10.00

The movie is recommended because it is an Animation Children's Comedy movie.
Animation is your top 9 preference
Children's is your top 6 preference
Comedy is your top 2 preference

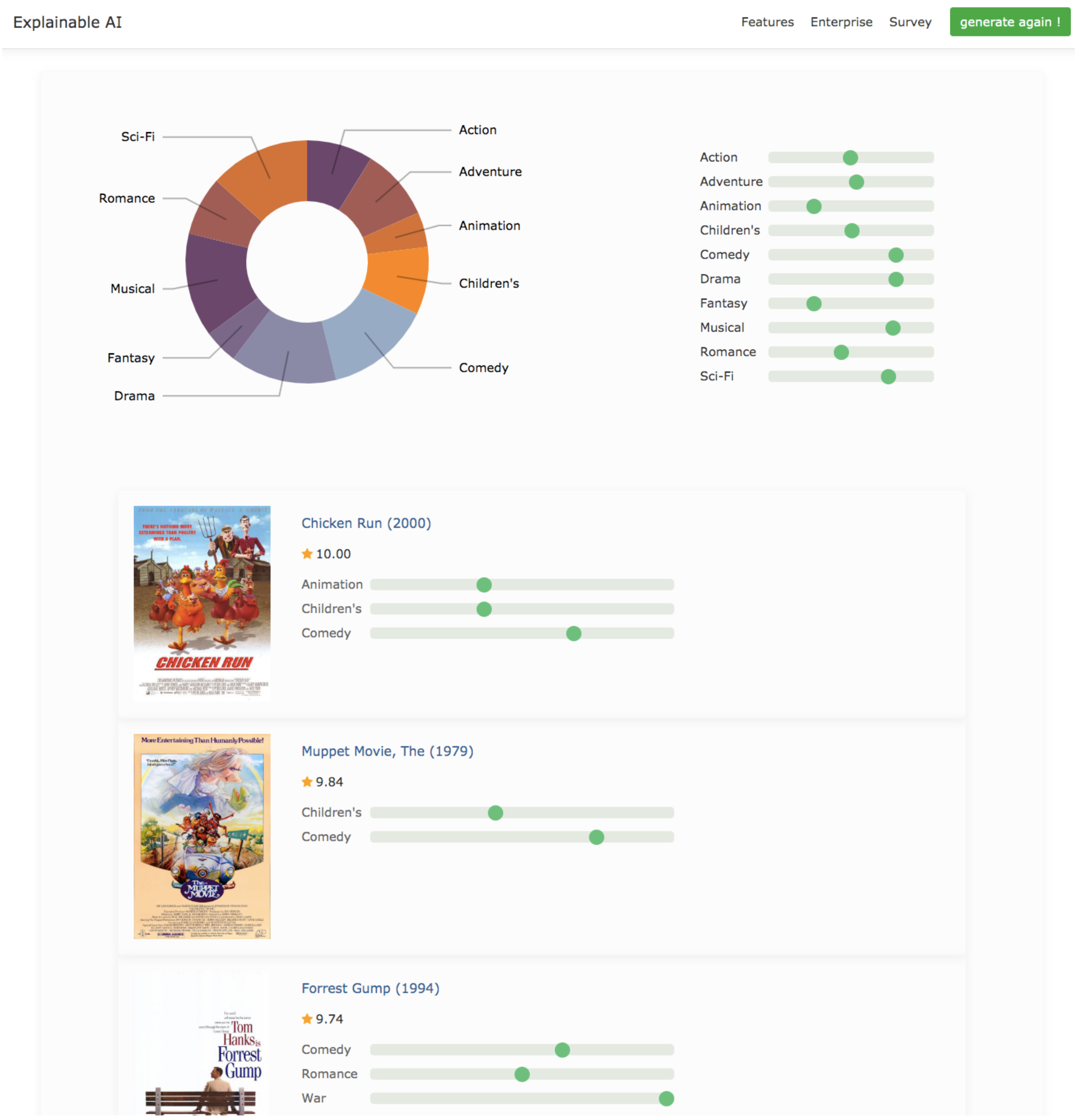
(b) textual explanation



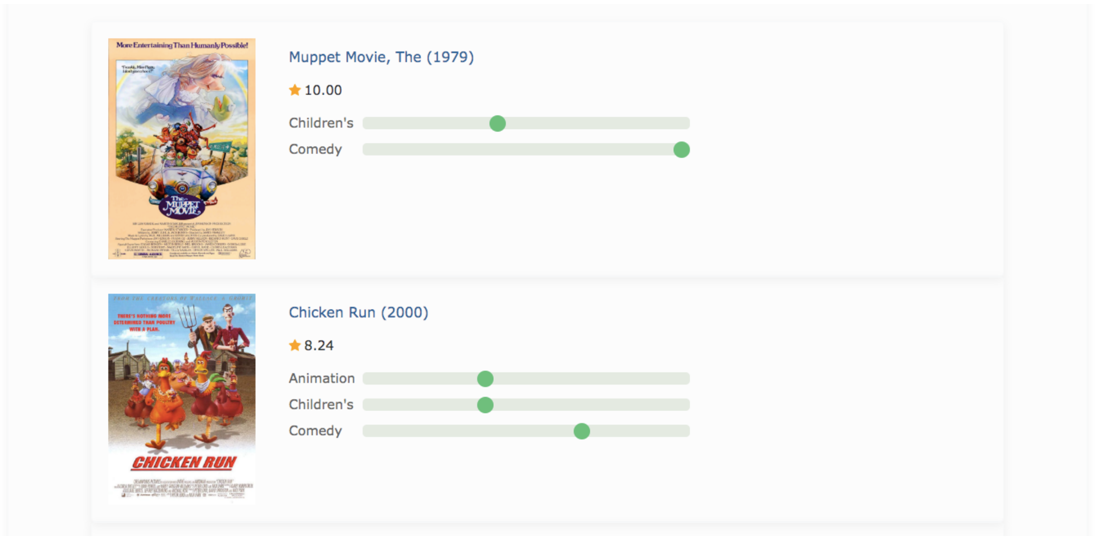
(c) no explanation

Figure 5.6: Different types of explanations

Figure 5.7 shows how to use sliders to explain the recommendations. By adjusting those sliders, user could update recommendations.



(a) Recommendations and explanations



(b) "The Muppet Movie" has a higher ranking after moving the slider of "Comedy" to the right

Chapter 6: Evaluation

6.1 Feedback Driven Development

Before talking about how we evaluate the system, firstly we would like to discuss how we managed the development process. During the development, we did not intentionally follow a particular software development methodology. But when we reviewed the entire process, the development is more evolutionary rather than sequential. We delivered a working software every week and discussed it with our supervisor. Our supervisor would evaluate the system and give some feedback. Those feedback would also refine the project requirements. Through multiple iterations, the software increasingly had more functionality and better quality.

In addition to getting feedback from our supervisor, in week 9, we also made a presentation about our system in front of several master and PhD students working on recommender systems. After the presentation, we let them evaluate our system and give some feedback. Some valuable suggestions we got from the discussion includes 'trying to implement multiple methods for displaying the explanations'. We adopted these suggestions and added 'using text to explain the recommendations' to the existing 'using sliders to explain the recommendations'.

6.2 Functionality, usability, and user experience

In 2006, McNamara proposed that when evaluating technology, there are three primary elements that need to be considered, namely, the product, the interaction between the user and the product, and the experience of using the product. Each of these three elements represents a unique but interdependent aspect of usage. These are Functionality (product), Usability (interaction), and Experience (user experience)[16].

6.2.1 Functionality

Evaluating functionality includes determining what features should be provided by the application. In other words, requirements should be properly satisfied by the application. Therefore, at the evaluation stage, we reviewed the requirements proposed at the beginning of the project and checked whether the requirements have been completed or not. Table 6.1 compares requirements and our actual implementation. From the comparison, we can know that our system satisfies the requirements.

Requirement	Implementation
presents movie recommendations	The system can present movie recommendations as shown in the Figure 5.5
presents an explanation with the recommendations	The system can present different types of explanation as shown in Figure 5.4
provides more than one explanation modality	The system can present numeric explanation using the sliders and present textual explanation.
provides a way to interact with the explanation	The user can change the explanation by adjusting the sliders.
provide a way to add new model	The user can add new model by uploading file to server
provide a way to gather evaluation data from end users	The system has a survey page which contains an online questionnaire. The administrator can update questions and gather user answers by connecting to the database.

Table 6.1: Requirements and corresponding implementations

6.2.2 Usability

Usability is related to how easy it is for the user to complete the task while using the application; the user experience focuses on the user's perception of how the application interacts with him. Although usability is different from user experience, they are both related to a user issue. Therefore, in this section, we discuss them together. To evaluate usability and user experience, the product needs to be tested with real users. However, considering the current situation, it is quite hard for us to evaluate the system with real users. Therefore, here we will only describe our plan of evaluating with real users.

Usability inspection is the name for a set of methods where an evaluator inspects a user interface. Generally, usability inspection methods can be classified into task-specific methods and holistic methods. Considering our system is mainly designed to complete a single task, which is movie recommendations, it is better to use task-specific methods to evaluate the interface.

The most representative task-specific method in usability inspection is called cognitive walkthrough. In Cognitive Walkthrough, a group of evaluators reviews the interface with a given brief description of the target user and are directed to take certain action sequences that eventually lead to a desired task to complete. Before the walkthrough, first we need to define inputs to the walkthrough. The inputs can be divided into four parts: users, task, correct action sequence and questions asked in the walkthrough notes. The cognitive walkthrough focuses on the performance of the interactive explanation among the explanation modalities as it involves the most interaction to the user interface. Participants performes the walkthrough are asked a set of questions for each subtask. These questions are used to determine the usability of the interface. In the case of our system, the intention of these questions is to clarify the following problems:

1. Can the user generate recommendations correctly without any instructions?
2. Can the user understand why these actions are necessary to generate recommendations?
3. Can the user understand why they need to adjust the sliders?

Table 6.2 shows the details of the walkthrough input.

Users	First-time visitors to the ACME system
Task	Get recommendation as well as interactive explanations
Correct action sequence	1. Enter user ID and click 'Submit' 2. Click 'Generate my recommendation!' 3. Click 'Recommendation with slider explanation' 4. Adjust the sliders 5. Click 'generate again!'
Questions asked in the walk-through notes	1. Will the user try to achieve the right effect? 2. Will the user notice if the correct action is available? 3. Will the user understand that the wanted task can be achieved by the action? 4. Will the user know that they have done the right thing after performing the action[17]

Table 6.2: Inputs to cognitive walkthrough

Chapter 7: Summary and Conclusions

This project addresses the problem of the explainability of recommendations. In the last semester, we conducted a literature review about the Explicit Aspect Model in order to develop the ACME user interface. This year we successfully implemented this interface. The interface can present movie recommendations along with an explanation. There are two explanation modalities and one modality allows the recommendation to be changed through interaction. The system also allows the administrators to upload new explanation models and gather user feedback. In conclusion, the core and advanced part of the project have been successfully completed.

As a software engineering project, the interface development has gone through requirement analysis, design, implementation and evaluation stages. During the development, we followed the agile principles by delivering the working software every week to sustain a very short feed loop and adaption cycle. Through this project, we also know how to apply the software engineering principles in actual development.

The interface has been presented to some master and PhD students working on recommendations systems. In the future, the system will be presented to Samsung in order to demonstrate ACME. The plan for the future is the system will go into production on some Samsung applications.

Acknowledgements

My deepest gratitude goes first and foremost to Dr. Neil Hurley , my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this report. Without his consistent and illuminating instruction, this thesis could not have reached its present form. Second, I would like to express my heartfelt gratitude to Dr. Tony Veale, who gave the essential instructions for how to structure the whole report. Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I also owe my sincere gratitude to my roommates who gave me their help and time in listening to me and helping me work out my problems during the difficult course of the report

Bibliography

[1]Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. ACM computing surveys (CSUR) 51, 5 (2018), 93.

[2]Francesco Ricci and Lior Rokach and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[3]Alexis Papadimitriou, Panagiotis Symeonidis, and Yannis Manolopoulos. 2012. A generalized taxonomy of explanations styles for traditional and social recommender systems. Data Mining and Knowledge Discovery 24, 3 (2012), 555–583.

[4]Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classier. In Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). ACM, New York, NY, USA, 1135–1144. https://doi.org/10. 1145/2939672.2939778

[5]Saul Vargas, Pablo Castells, and David Vallet. 2011. Intent-oriented diversity in recommender systems. In Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011. 1211–1212. https://doi.org/10.1145/2009916. 1151 2010124

[6]N. Hurley. ACME: ACME: AspeCt Model Explainer for Top-N Recommenders, Unpublished papers, private communication with N. Hurley.

[7] Q. Liao. Diverse Recommendations under the COVer. FYP Report 2018, UCD, School of Computer Science.

[8] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. arXiv preprint arXiv:1812.04608, 2018.

[9]Lawshe, C.H. (1975). A quantitative approach to content validity. Personnel Psychology, 28, 563–575. doi:10.1111/j.1744-6570.1975.tb01393.x

[10]Johnson-Laird, P.N. Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness. Cambridge University Press (1983).

[11]Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. 2012. Tell me more? the effects of mental model soundness on personalizing an intelligent agent. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12). Association for Computing Machinery, New York, NY, USA, 1–10. DOI:https://doi.org/10.1145/2207676.2207678

[12]Babu Thomas, movielens-posters, GitHub repository, https://github.com/babu-thomas/movielens-posters

[13]Wurman, Richard Saul. Information Anxiety. New York: Doubleday. 1989.

[14] Hull, Elizabeth, Kenneth Jackson, and Jeremy Dick. Requirements Engineering, London: Springer, 2005.

[15] https://www.mongodb.com/compare/mongodb-mysql

[16]Niamh McNamara, Jurek Kirakowski. Functionality, usability, and user experience: three areas of concern. interactions Volume 13, Number 6 (2006), Pages 26-28

[17]Wharton, C., Rieman, J., Lewis, C., Polson, P.G. (1994). The Cognitive Walkthrough Method: A Practitioner's Guide. In J. Nielsen R. Mack (Eds.), Usability Inspection Methods. (pp. 105-140). Wiley.