Integrated contents recommendation platform across different domains

Jihyeong Lee, 2017314320
Chanhoo Keum, 2017314910
Inseo Nam, 2019312643
Seoyoon Hong, 2018310346
Jinhwa Hong, 2017310820

Sungkyunkwan University Capstone Design Project

Abstract. Many contents such as movies, dramas, and webtoons are scattered in various domains. We have the inconvenience of having to move the domain to see each content we want, and there is no recommendation system among them, so we don't know which content suits our interests. In this project, various contents will be included in one service, and a recommendation system will be implemented based on these contents with contents based and collaborative filtering model.

Keywords: Cross Domain Recommendation · Contents-based Recommendation · BERT · RecBole · Web Service

1 Introduction

These days, lots of platforms including OTT(Over The Top) services uses state of art AI recommendation systems. And there are also several platforms which provide targeted recommendations based on ratings from the users and similar contents for each content.

However, there are some limitations from user's view. With OTT platform that provides actual contents, user can be provided personalized recommended contents from various domains(e.g. movie, drama, documentary) but the recommendations are limited within the platform, which means user cannot get recommendations of other contents that does not exist in the platform one is using. For example, OTT service domains that we often use include Netflix, Watcha, and Disney Plus. Of course, there are overlapping contents, but the types of contents are very different for each domain due to exclusive content, etc., and it is practically difficult to integrate them and provide recommended services at once.

And in case of rating services, they do not provide cross-domain recommendation and a user can get recommendations only after when one gives the rating for several contents for each domain, e.g. books, movies, drama. Although we do not provide services so that we can show movies right away or show webtoons directly, we will be able to integrate many contents scattered for each domain to apply the recommendation system to reflect user preferences. Realistically, this service will not only provide users with solutions to light problems such as which OTT should be used, but also provide users with a glimpse of domain integration content based on individual preferences. They cannot get personalized recommendation in domains they have left their ratings and not in the other domains.

Therefore, we are planning to develop a service that provides cross-domain recommendations with fewer ratings which is not restricted to any content-providing platforms. We are expecting that a user can get list of similar contents of one content and targeted recommendations not only from one domain but also from other domains, even if he or she have left ratings in only one domain. This service also could be a good tool for content providers to understand the needs of users given the current market trend where developing cross-domain contents become active more and more.

2 Design

2.1 Overall Architecture

The goal of this project is to recommend other content that is judged to be the most similar to the user's preferred content. Users will be able to receive recommendations on topics similar to their preferred content through the website. To this end, membership registration is carried out on the website. During the membership registration process, it goes to the preference survey page and allows users to select their preferred content. It is a system that recommends other similar content based on the selected content when logging into the website after all procedures are successfully completed.

2.2 Frontend

Make sure that the main page is displayed first. The content that appears in the initial state will not be related to the user's preference because the user has not yet entered the preference through membership registration or login. The membership and login buttons are located at the top, and you can log in and sign up through a new window or a modal window through each button. In the case of login, only one window is required, but in the case of membership registration, there is a total of two windows, one window for entering basic information such as the user's personal information and one for conducting preference surveys. When this process is completed and login is carried out normally, it can be seen that the recommended content is different based on the preference entered, unlike the content that was initially viewed.

2.3 Backend

The project is not to directly show the user a movie or webtoon, but to recommend other content similar to the entered content. Therefore, we crawl on a total of four types of content, including many, various movies, dramas, webtoons, and books. In addition to information such as the name, information, and genre of the work, the picture of the work is crawled so that the work can be recommended with photos and posters when recommending content in the future. This information is stored using SQL and linked to the AWS server so that data can be uploaded or downloaded.

2.4 AI

We aim to provide detailed information on each work and a list of similar works, as well as personalized recommendation lists for three different domains once the user has left ratings for a few works. Through this, users will be able to receive recommendations for works that match their tastes across various domains, in addition to content-based recommendations.

There are movies, dramas, and webtoons in the three domains. These are all things that modern people like and see often. We will apply the recommended system for these three domains. Users receive services in the form of websites. First, you sign up for a membership, and there is a window for entering personal content preferences during the procedure, which will determine what content the user likes. It then learns the preferences and displays the relevant content on the main page of the website. Click on the content to view the details and like it You can also mark it. This behavior will allow servers and AI models to continuously judge users' preferences and provide more accurate recommendation services.

3 Implementation

3.1 Personalized Recommendation

The user will receive a recommendation system service based on the preferred content selected at the time of membership registration. When signing up for a membership, it was set to complete the membership only when a specific number or more must be selected. And when you log in, you will recommend movies, dramas, books, and webtoons of similar genres on the main page based on the selected content.

When you click on each content, it goes to a window containing a description of the content. Information such as photos, titles, genres, and writers of the content will be displayed. And if you go to the bottom, some content similar to the clicked content is recommended again. There is a like button under the picture. Clicking this button saves the clicked content to My Page. If you go to My Page, you will see all the content that you pressed the Like button at a glance. If you delete a like, it will also be deleted from My Page. And content search through keywords is also possible. Content can be searched through specific words, and you can search by the name of the work, or you can search by writer or actor. If there is no result through the search keyword, the most similar content is displayed.

3.2 Dataset

In order to more accurately recommend similar content, the number of content that you have must be large. In that case, more learning can be carried out and the accuracy of the association is increased, enabling more accurate recommendations to users. To this end, movie data was crawled on Watchapedia. In the case of dramas, Watchapedia crawled, and webtoons crawled webtoons existing on Naver webtoon sites. In addition, it is the same literary work, and as part of the cross domain recommendation, book information was crawled, and it helped learn similarity a little more. Finally, about 22,000 pieces of data were collected, and about 10,700 pieces of data were actually used. Through this, it is possible to create a model with a slightly higher similarity.

3.3 Model

We have selected a BPR model for actual service, considering performance and inference speed. BPR focuses on pairwise comparisons of items. During training, it samples pairs of items and user interactions. For each pair, BPR learns to predict whether the user prefers one item over the other based on their interactions. The goal is to learn embeddings that rank preferred items higher than non-preferred items for each user.

3.4 Search

Dataset For training, we used KorSTS dataset provided by KakaoBrain. it has 8,628 pairs of Korean sentences and similarity score between them. It is composed of 5,749 pairs of training data, 1,500 of validation set, and 1,379 of test data.

Model We used pretrained sentence Bert model for content-based filtering. Unlike the original BERT, it goes one step further to encode the semantic meaning of the input sentence. Since the similarity is calculated with those embedding vectors, it is much more efficient than original BERT. The original BERT gets two sentences and 'SEP' token as input, and returns the similarity. To calculate the similarities between n sentences, the model should run for nC2 times. On the other hand, Sentence-BERT calculates the similarity with embedded vectors, the model should run only n times. Total Similarity = 0.2*Title Similarity + 0.1*Genre Similarity + 0.7*Description Similarity

The configuration of model follows BERT-base model, except the maximum sequence length which was set to 256 because we do not have long input sequence. Mean pooling was used to get the embedding vector. Learning rate was set to 1e-5, and MSE Loss and AdamOptimizer were used. The model was fine tuned for 4 epoch, and the MSE score and R-squared score of test data was 0.036 and 0.558.

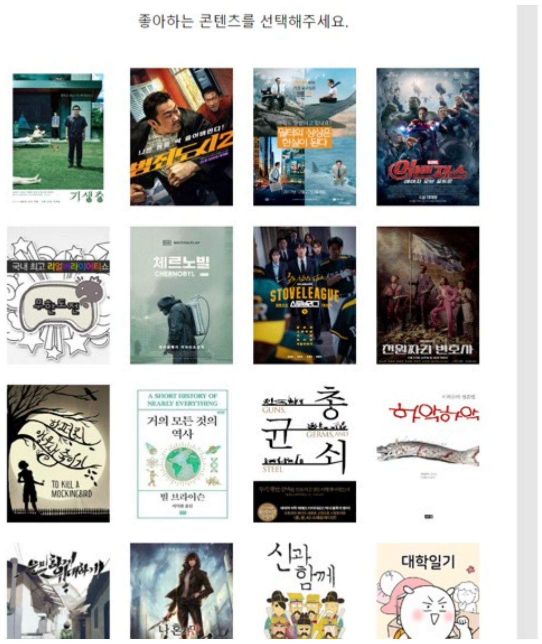

Image 1: Preference(in Register) Page

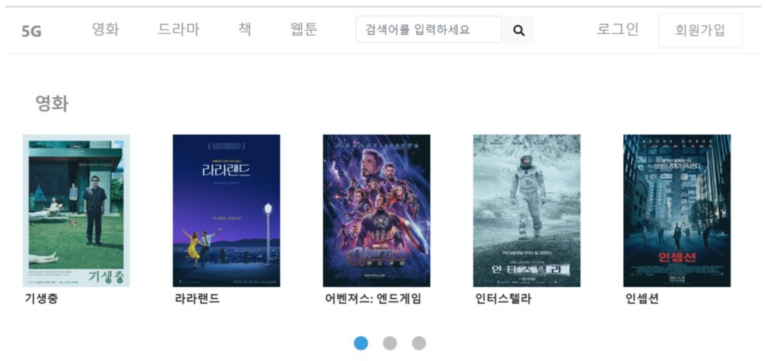

Image 2: Index(Main) Page

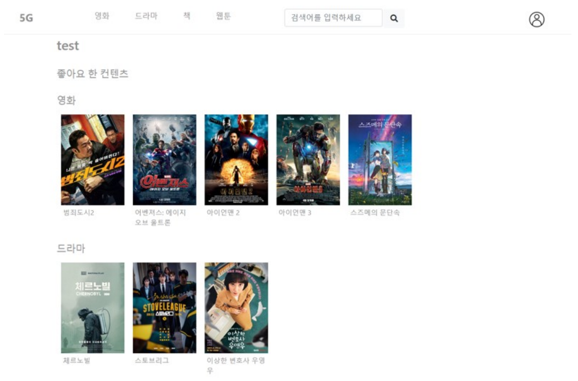

Image 3: My Page

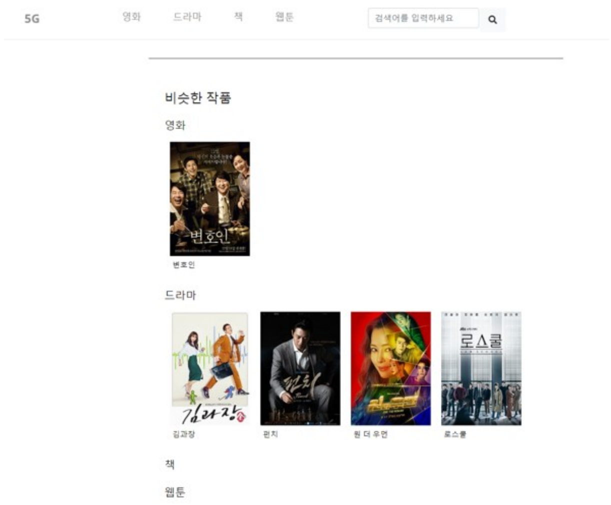

Image 4: Content Page

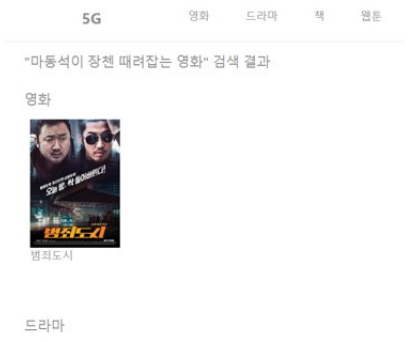

Image 5: Search Page

3.5 UI/UX viewpoints

When you access the website, the main page appears first. In the upper left corner, there is a main button and a button that can be moved to the movie, drama, and webtoon sections. There are login and registration buttons on the upper right and pop up in the form of a modal window. If you are not logged in, the most recent or most popular content will appear in each section. After logging in, the user's ID and access token are stored in cookies, and when accessing the main page, personalized content based on individual preferences will be displayed. When a user clicks on one content, they can access a detailed information page. Within this page, they can click on the ”Like” button to update their preference information, which will be utilized in subsequent model training. Users can perform searches and navigate to the search page to view the search results.

3.6 Frontend details

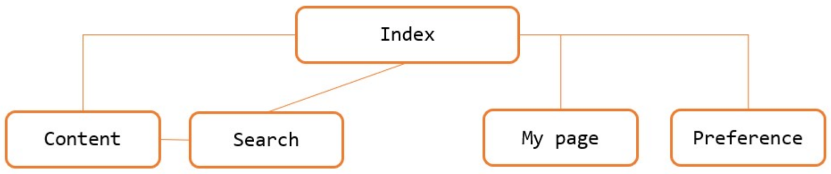

Image 6: FrontEnd Structure

There are a total of five pages in Front End (FE). First of all, Index is the main page to provide the recommendation system, which is this project. The page recommends movies, dramas, books, and webtoon contents based on user preferences. If you click on the content, it will automatically go to the Content page. This page shows a description of the content. It shows the title, writer, genre, etc. along with content photos. There is a like button under the picture, which allows you to save content to My Page. Scroll down to show another content similar to the content on the current page. The search proceeds to the search page, which shows the results of the keyword search. Contents including keywords searched by title, writer, actor, etc. will be shown. Like Index, clicking on the content will take you to the Content page. All features were implemented through HTML, CSS, Javascript and JQuery.

There is a My Page where you can check the content that you like at a glance. Like it on that page If you click on the content, it will also go to the Content page, and if you cancel the like, it will disappear from the My Page. There is also a Preference page that conducts preference surveys. Here, the user's preference is investigated, and content recommendation is made by implementing an optimal recommendation system based on this.

3.7 Backend details

API This project uses FastAPI. FastAPI has the following characteristics.

FastAPI is built on top of Starlette, a high-performance ASGI server. This allows to deliver fast request-response cycles and handle high loads efficiently. And it leverages Python's type hints extensively. By using type hints, you can improve code readability and enable better static type checking and debugging. And it automatically generates interactive API documentation using Swagger UI or Redoc. It provides a detailed view of your API endpoints, request and response models, parameters, and more, making it easier for developers to understand and consume your API. And it integrates with Pydantic, a powerful library for data validation and serialization. It allows you to define data models with field types, validation rules, and serialization capabilities. Plus, it automatically validates incoming requests against these models and handles serialization of responses. And it fully supports asynchronous programming using Python's async and await syntax. And it includes security features out of the box. It supports various authentication methods, including OAuth2, JWT (JSON Web Tokens), and more. It provides convenient decorators and tools to secure your API endpoints. Finally, it has built-in support for WebSocket communication.

Through this, we were able to facilitate the interworking of databases and SQL. FastAPI provides various methods to perform CRUD operations on databases. Therefore, in this project, we used FastAPI for interworking with MySQL through FastAPI.

Database We use mysql database. To manage the unified index of all contents, we have adopted the following approach due to separate management of database tables for each category. The file "contents_idx.json" contains information about the titles and categories (movies, dramas, books, webtoons) of each content. Since all the titles and categories of the contents in the database are stored in this file, we can manage the integrated index here. When information about a specific content is requested, we search for the title and category of the content in the "contents_idx.json" using the index. Then, we access the corresponding database based on the category and perform a search using the content's title to return the results.

We used Lightsail instance called LAMP_PHP_8-1 of amazon web service(aws) as a hosting server. We uploaded all of our frontend and backend system by cloning git repository inside the server. When running uvicorn app, we used the public IP of the server as a host, and designated port, so that we can access the website using http address from other computers(clients). Also we used Mysql as our database inside the server so that backend systems interact with the DB inside the server. We created and managed our DB tables using phpmyAdmin of the server. We uploaded user information and contents data(csv files) into the DB.

Deployment We used Lightsail instance called LAMP_PHP_8-1 of amazon web service(aws) as a hosting server. We uploaded all of our frontend and backend system by cloning git repository inside the server. When running uvicorn app, we used the public IP of the server as a host, and designated port, so that we can access the website using http address from other computers(clients). We used Mysql as our database inside the server so that backend systems interact with the DB inside the server. We created and managed our DB tables using phpmyAdmin of the server. We uploaded user information and contents data(csv files) into the DB.

4 Evaluation

The following Table 1 shows metrics used in the experiments. They are performance indicators commonly used in recommendations. Recall is the true positive divided by the sum of true positive and false negative. Precision is the true positive divided by the sum of true positive and false positive. MRR and NDCG are metrics specifically used in recommendations and were used to include the recommendation order in the performance evaluation. If five contents are recommended, even if all five are included in the contents the user has seen, a model that recommends preferred items first is better, so a weighting method is used based on the recommendation order. Hit represents the proportion of users who have at least one hit item among all users. In terms of performance metrics, ItemKnn showed the best performance. However, considering the inference time, ItemKnn took a significantly long time, which is not desirable for actual service deployment. Therefore, we selected BPR as the final model, which has a shorter inference time.

	recall@10	mrr@10	NDCG@10	Precision@10
Pop	0.0379	0.2585	0.1008	0.0859
BPR	0.0409	0.0.2745	0.0.1123	0.0.0976
ItemKnn	0.059	0.3811	0.1655	0.1404
NeuMF	0.0408	0.2598	0.1073	0.0951
MultVAE	0.0524	0.3204	0.1379	0.1199

Table 1. Evaluation metrics and scores

5 Related Work

5.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a groundbreaking language model developed by researchers at Google. It belongs to the family of transformer-based models and has revolutionized various natural language processing (NLP) tasks.

Unlike traditional language models that process text sequentially, BERT uses a bidirectional approach, meaning it considers both the left and right context of each word during training. This bidirectional nature enables BERT to better capture the contextual relationships between words and understand the nuances of language.

5.2 Sentence BERT

Sentence-BERT (SBERT) is a framework for generating fixed-size dense vector representations (embeddings) for sentences or short texts. It is based on the idea of pretraining a model on a large corpus of sentence pairs and then finetuning it for specific downstream tasks such as sentence similarity, paraphrase detection, and clustering.

The key innovation of Sentence-BERT is its use of siamese and triplet neural network architectures during pretraining. In the siamese architecture, two copies of the model share the same weights, allowing them to encode two input sentences simultaneously. The model is trained to generate similar embeddings for semantically equivalent sentences and dissimilar embeddings for sentences with different meanings. This encourages the model to learn a meaningful representation of sentence semantics.

5.3 RecBole

We used Recbole for training and evaluating our models. RecBole is an open-source recommendation framework developed by researchers from the National University of Singapore. It aims to provide a flexible and efficient platform for developing and benchmarking recommendation algorithms. RecBole supports a wide range of recommendation tasks, including personalized recommendation, sequential recommendation, session-based recommendation, and more. At its core, RecBole provides a modular and extensible architecture that allows researchers and developers to easily implement, compare, and reproduce different recommendation algorithms. The framework is built on top of PyTorch, a popular deep learning library, which provides a flexible and efficient computational graph for training recommendation models.

6 Limitations and Discussions

When implementing and demonstrating functions locally and on the server, it worked normally, but this process was the result of one person at a time. As a result of two or more people accessing the server at the same time and conducting the test, the recommend file was not written well. When multiple users access at the same time, a solution is needed so that appropriate processing can be performed. And depending on the category, there were cases where the recommendation of works was not properly made due to the significantly small number of categories. If there are significantly fewer items in the category or if they are not at a significant level, filtering should be carried out and learning should be conducted more strictly. In addition, it took about 3 minutes to learn from the selected content after signing up as a member. Considering the general work service or content, 3 minutes is considered to be quite a long time. I think this time can be greatly shortened if we run a website and conduct learning on a GPU-based server for artificial intelligence models. And also it was not possible to obtain an evaluation or feedback on the service quality or performance of this project because it was not possible to cross-check with other groups. If the service can be provided in the future, we will receive help from various users to improve performance through experience.

7 Conclusion

In conclusion, we have proposed the development of a cross-domain recommendation system that aims to address the limitations of existing platforms by providing personalized recommendations across multiple content domains. Our system integrates content from diverse domains and leverages user preferences to offer targeted recommendations. We have presented the architecture, design, implementation, and frontend-backend details of the system, along with the dataset, model, and search functionality utilized in the recommendation process. Additionally, we have discussed the UI/UX aspects, supported by visual representations of the frontend structure. For the backend, we have employed FastAPI for API development and MySQL for database management, and the system is currently deployed on an AWS Lightsail instance. In summary, our paper provides a comprehensive overview of our proposed cross-domain recommendation system and its implementation.