LUDWIG-MAXIMILIANS-UNIVERSITÄT AT MÜNCHEN
Department “Institut für Informatik”
Lehr- und Forschungseinheit Medieninformatik
Prof. Dr. Andreas Butz

Master’s Thesis

Explanation Adaptation for Hybrid Multimedia Recommendation System

AUTHOR
EMAIL

Bearbeitungszeitraum:	START bis END
Betreuer:	SUPERVISOR
Verantw. Hochschullehrer:	Prof. Dr. Andreas Butz

Aufgabenstellung

THESIS TITLE

Problem Statement TO BE ADDED

Scope of the Thesis TO BE ADDED

Tasks TO BE ADDED

Requirements TO BE ADDED

Keywords TO BE ADDED

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 8, 2020 . . . . . .

Acknowledgments

I would like to appreciate ...

Abstract

This thesis proposes ...

Contents

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 LaTeX

The first release of LaTeX [Lamport, 1994] appeared in 1983 [Lamport, 2007].

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

the related-work part 1.

2.1.1 conversational RS and Hybrid RS

2.2 Explanation Recommend System

the related-work part 2.

2.3 Adaptation for ExplanationRecommend System

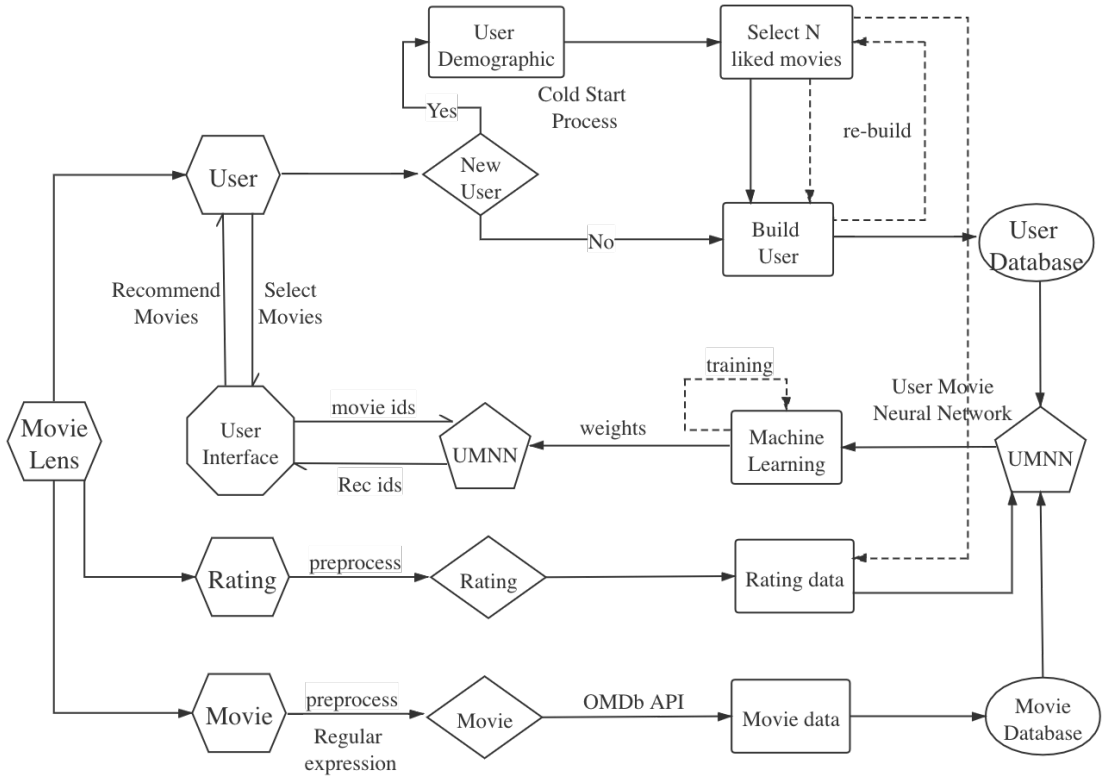
the related-work part 3.

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.

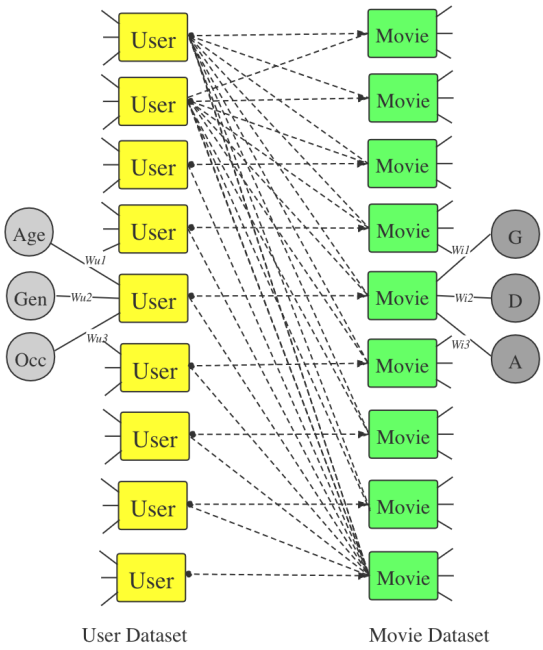
In terms of user trajectory, each time a user comes in, it is necessary to determine whether the user is a new one. Once a new user is found, a cold start strategy will be initiated. The system will guide the user to enter relevant personal information (gender, age, occupation). Then take a series of 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 UMNN

3.1.4 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 build-

Figure 3.3: frontend and backend

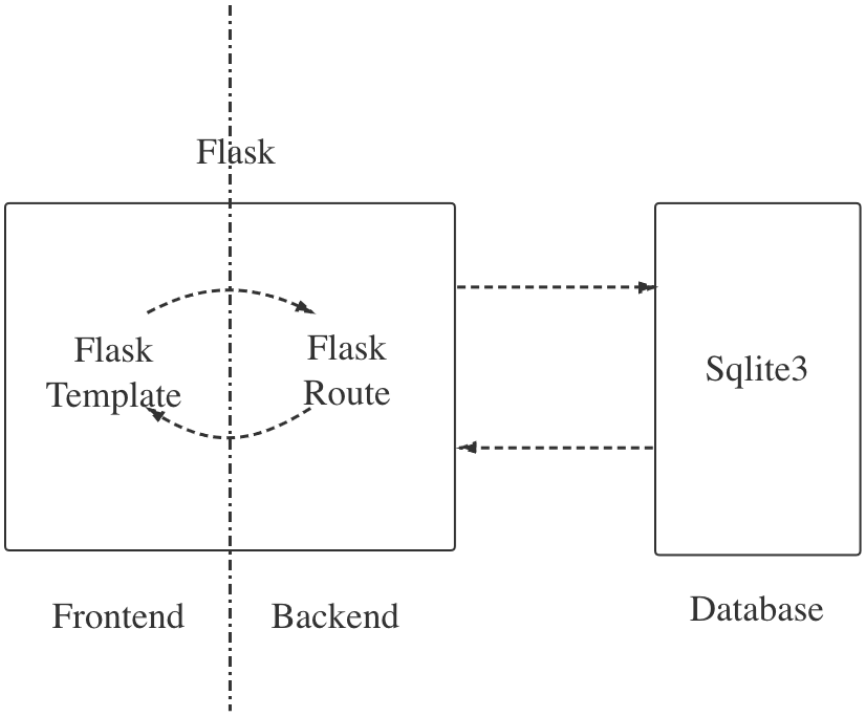
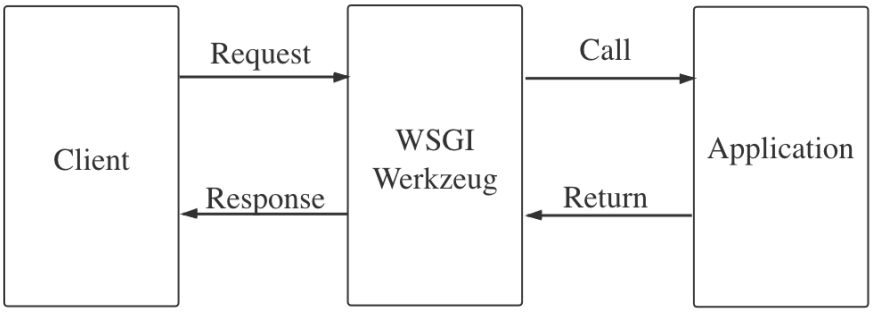


Figure 3.4: Flask working process



ing 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.

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

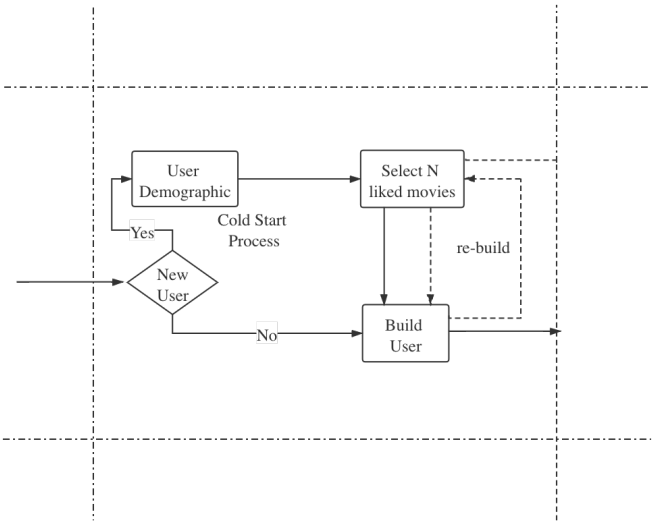
3.1.5 Cold Start

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:

Figure 3.5: Cold start process



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

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.

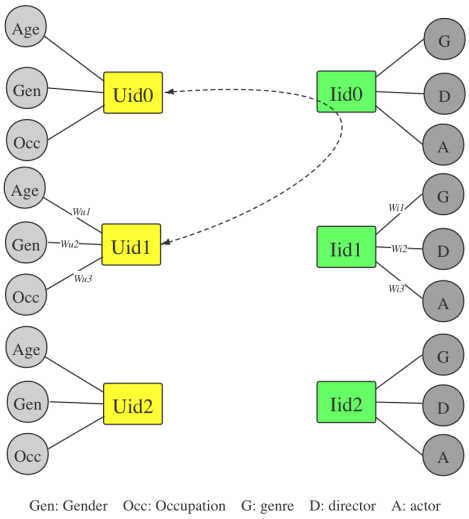
3.2 Recommendation Strategy

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

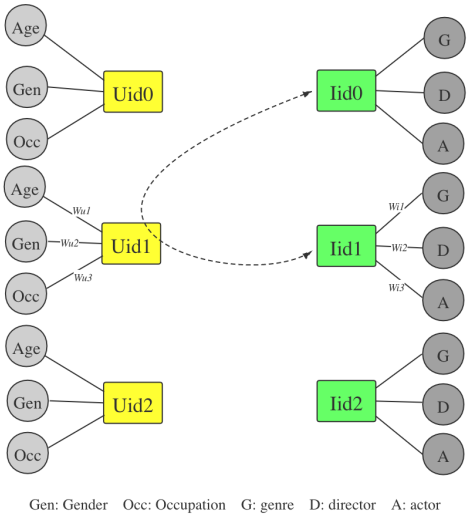
Figure 3.6: user-based recommendation strategy



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

Figure 3.7: item-based recommendation strategy



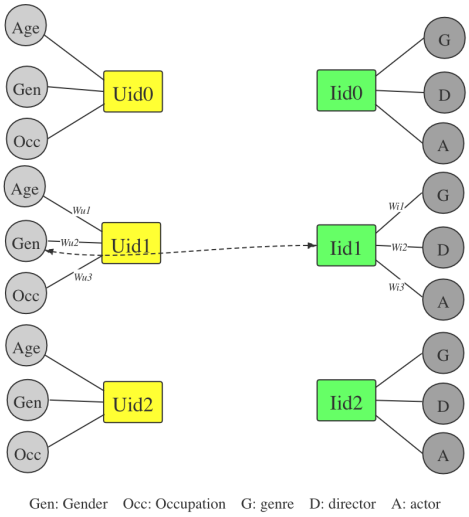
current behavior. The similarity between the item recommended to the user and the item previously selected by the user is understandable by the user, which is called Interpretable. Most of the recommended items are not popular, but are related to the interests of users. This recommendation method has 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

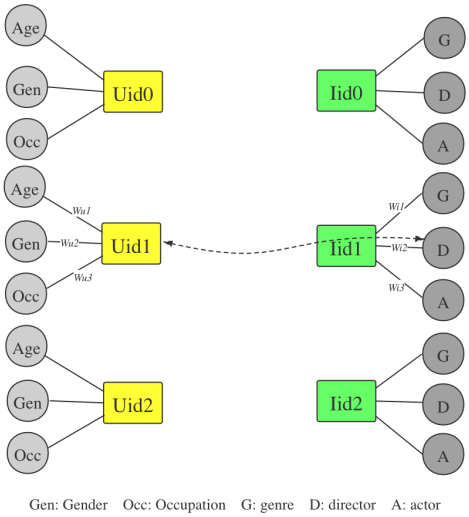


and then make recommendations. At present, it is rarely used alone in large systems, and it

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 she 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

3.2.3 Recommendation Explanation Adaptation Strategy

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 Neural Network Model

3.3.3 Training Weights

3.4 User Interface Prototype

3.4.1 Development Tool and Language

3.4.2 Prototype

3.4.3

3.5

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 Data Set

4.2 User Study Process

4.3 User Study Metric

4.3.1 Confidence

4.3.2 Transparency

4.3.3 Satisfaction

4.3.4 Persuasiveness

4.4 Result analysis

5 Evaluation

> 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

6 Application

> 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

7 Discussion

> 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

8 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

9 Appendix

> 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

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.

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

[Lamport, 1994] Lamport, L. (1994). *LATEX: a document preparation system: user's guide and reference manual*. Addison-wesley.

[Lamport, 2007] Lamport, L. (2007). The writings of leslie lamport: Latex: A document preparation system. *Avible: Leslie Lamport's Home Page. Retrieved*, pages 04–27.

[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.

[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.