

A Real Time Movie Recommendation System

Qiyu Zhang (qz5uw)

Abstract—In this project, I developed a real time movie recommender system which contains a website and Alexa sound device. For the website, it's based on Flask framework, applying collaborative filtering algorithm to recommend users their potential liked movies. Here I used S3 as a data storage not only to store users and movies' information but also for easy accessing data and then training the model. And the recommended movies will be stored in a DynamoDB table for each user as a cache, so next time the same list will be generated in a shorter time. It's deployed in AWS EC2 and I use Gunicorn and Nginx for a better performance. For the echo, it's developed and deployed in Alexa and Lambda was used as the endpoint. To connect it with the website and serve users better, it could visit the DynamoDB to read the data directly.

Index Terms—Recommender, AWS, Echo

I. INTRODUCTION

A recommender system or a recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. [1] For the movie recommender, there are some companies keeping working on this, like Hulu, Netflix and so on. And typically there are four methods, content-based, collaborative filtering, matrix factorization and deep learning. [2] The Collaborative Filtering Recommender is entirely based on the past behavior and not on the context. More specifically, it is based on the similarity in preferences, tastes and choices of two users. It analyses how similar the tastes of one user is to another and makes recommendations on the basis of that. In general, collaborative filtering is the workhorse of recommender engines. The algorithm has a very interesting property of being able to do feature learning on its own, which means that it can start to learn for itself what features to use. [2] And this is the algorithm I applied to this project, we will talk about this later in Part 4 Algorithm.

Here, I also used some tools in AWS. Using cloud computing to develop and deploy application has flexibility and efficiency. [3] For example, it's not necessary to store the data on the virtual machine. Instead, I used S3 and DynamoDB to hold data which also offers easy access for my Echo. The Internet of Things (IoT) represents a new class of applications that can benefit from cloud infrastructure. [4] I bought an echo on black Friday and thought it's very interesting. So I extended the system to Alexa. For the details of developing tools. We will talk about them in the next section.

II. ARCHITECTURE

A. LOGIC

This project contains website application and echo application. The logic graph is shown in Fig. 1. Because this is a personal project and some other reasons, I didn't include the

"create user" and "Add movies". In the front page, you just need input your id and the amount of movies that you want. There are several logic steps. First let's talk about the website.

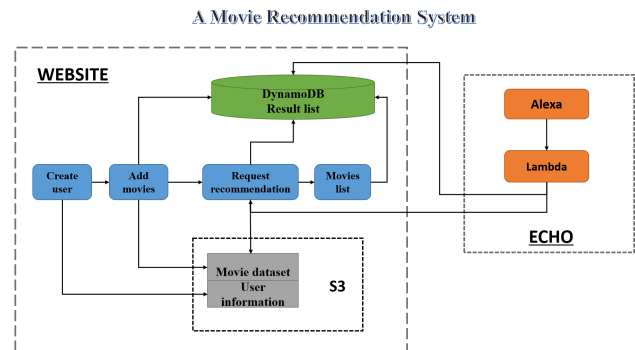


Fig. 1. Design of this movie recommendation system

- Step 1: if you are a user, you need create your profile. The person information will be store in the user information file in S3
- Step 2: For getting recommendation, the user should add the watching experience. And the movie information will be stored in the movie information file in S3
- Step 3: After updating the watching experience, it will trigger training model part because the personal movies have been changed
- Step 4: When requesting recommendation (train the model), it will visit the S3 for user information and movie information to train or retrain the model
- Step 5: After generating a movie list, it will not only show this list in the front page, but also store the list in the DynamoDB as a cache
- Step 6: Next time if user request some movies without changing watching experience, the recommended list will be generated from the record already in the DynamoDB.
- Corner case: If the user input an invalid id or amount number, it navigates to the error page and remind user try again

For Echo:

- Step1: When user talk with echo, the lambda handler will be triggered and return back the different words depends on questions
- Step2: When echo is asked for recommending movies, echo will firstly check whether there's record exist in the DynamoDB table. If it is, echo will give back the list. If not, Alexa will ask user to add watching experience in the website for the first step

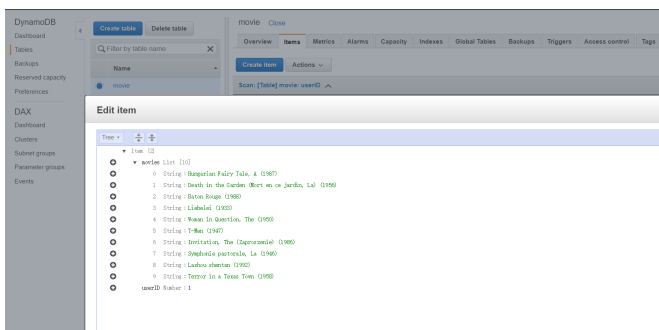
- Corner case: If the user input an invalid id or amount number, it could navigates user to try again. And the capacity for each user in DynamoDB table will be at most 10. So if user ask for a very big amount movies, it would just return back 10 movies.

We will talk about why we use such tools in the next subsection

B. Development Tools

1) *Website*: For the website, I used Flask as a framework. Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.[5] So it's very convenience for the small project like this one and easy to start. In the front end, to make the page better looking, I utilized bootstrap. Bootstrap is a free and open-source front-end framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many earlier web frameworks, it concerns itself with front-end development only. [6] Because there's no too much interaction between front and back end, so bootstrap is a good choice.

2) *Storage*: For the storage, instead of put the data in the local machine, I used 2 storage service S3 and DynamoDB. There are movie data file and user data file storing in S3. Because the system may need to visit these files for training model. If the data are in the database, it's slow to read all the data. Also, S3 is very stable to store the most important data. DynamoDB is a fully managed proprietary NoSQL database service that supports key-value and document data structures. [7] The previous movie lists could be stored here for different users. So if the user requests recommendations for many times without changing the watched movies, the result will not change. In that way the system could return back the previous list directly to save time. The storage example is shown in Fig. 2.



Item ID	movie
1	Shogun: Regency Palace Tale, 4 (1987)
2	Shogun: Death in the Garden (Short on the Garden), 14 (1988)
3	Shogun: Berlin House (1988)
4	Shogun: Labeled (1988)
5	Shogun: When in Question, The (1988)
6	Shogun: T-Rex (1987)
7	Shogun: Sentimental, The (Operational) (1988)
8	Shogun: Shephard's journey, 14 (1988)
9	Shogun: Ludlow's journey (1988)
10	Shogun: T-Rex 14 x Texas Town (1988)

Fig. 2. item in dynamoDB table

3) *Deployment*: To deploy this web application. I used an EC2 to run it. However, instead of running is directly, I used Virtualenv, Gunicorn and Nginx for better performance. virtualenv is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary

executables to use the packages that a Python project would need. [8] The Gunicorn is a Python Web Server Gateway Interface (WSGI) HTTP server. And it is a pre-fork worker model. [9] Nginx is a web server which can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. [10] So it will be safe and efficient to use Nginx. The deployment layers logic is shown in Fig. 3.

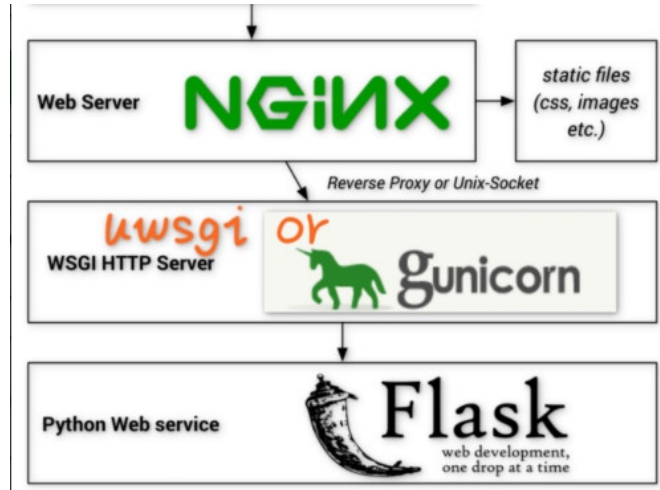


Fig. 3. deployment layers

4) *Echo*: Amazon Echo is a brand of smart speakers developed by Amazon. The device supports voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic and other real-time information. It can also control several smart devices acting as a home automation hub. [11] Because it belongs to amazon's product, so it's easy to interact with other tool that I mentioned above. To control it, It's not necessary to let it run all the time. So Lambda would be a good behavior handler. The Lambda model has many benefits as compared to more traditional, server-based approaches. Lambda handlers from different customers share common pools of servers managed by the cloud provider, so developers need not worry about server management. The Lambda function is shown in Fig. 4.

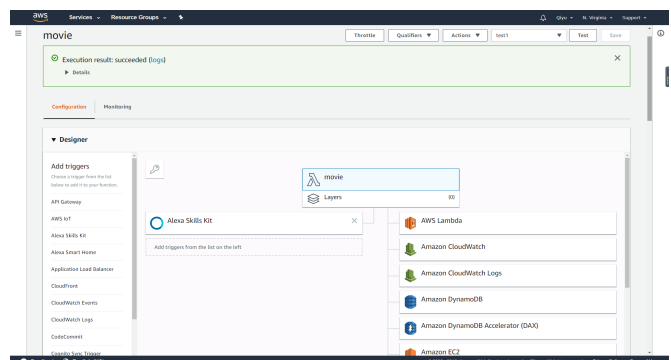


Fig. 4. Lambda for Alexa

C. ALGORITHM

Although this project mainly focus on development by cloud tools, I will talk a little about the data set I used and the collaborative filtering.

1) *Dataset*: Dataset is called MovieLens which contains 100K, and I did some preprocessing to select the useful data.

2) *Collaborative Filtering*: collaborative filtering is based on User Similarity and the follows the steps below

- Build the user-item Rating Matrix
- Build the Distance Matrix based on Cosine similarity
- Predict the rating based on Distance Matrix by Weighted Average

III. RESULT

A. Test of Web Application

The website contains 5 pages, Home, Movie, About, Result and error page. Movie page is where we can submit the form containing user id and amount number. And then the result could be shown in the Result page. Below shows the Movie and Result page.

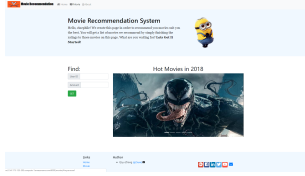


Fig. 5. movie page to submit form

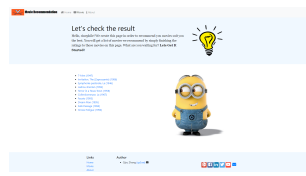


Fig. 6. result page to show list

If there's an error like user id is smaller than 0, then error page will appear and navigate you back to the Movie page.

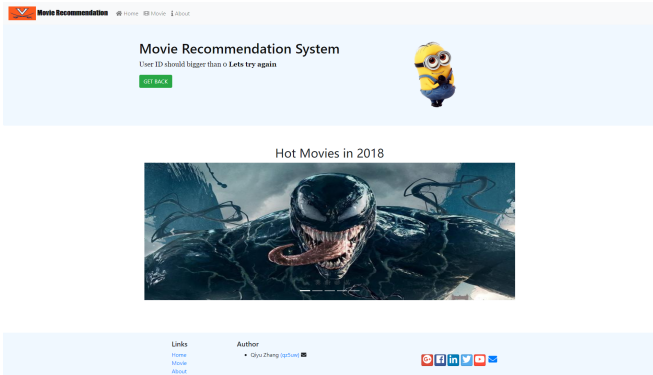


Fig. 7. error page

More details could be visited in my github. <https://github.com/QiyuZ/movie-recommendation-web>

B. Test of Echo

For testing of Echo, it's hard to show here. So I upload it on youtube. Please visit the following link <https://youtu.be/yzSNARaMHXk>

And there are some if-conditions in the Alexa design as shown in Table 1.

TABLE I
COMMUNICATIONS BETWEEN USER AND ECHO

user	case	echo
open movie	always	"Welcome to the movie recommendation system. what can I do for you?"
request movies	always	"Ok cool. Please tell me your ID and how many movies do you want?"
submit id and amount	record exist in database	return back a list of movies
submit id and amount	record does not exist in database	ask users to visit web firstly.
submit id and amount	record exist but requested amount is large than 10	just recommend 10 movies
stop	always	"Thank you for using the movie skill. See you next time!"

IV. DISCUSSION

To sum up, this project realize to recommend movie both in the website and Alexa in real time. To optimize the performance, firstly, I used S3 to store the user and movie data for training model. Also it takes DynamoDB as the database to store recommended movies for each user. However, I also think there are some parts in the project that could be improved

1) *More functions*: To make this application completely real time, the two parts "create user" and "add movies" are necessary. Because of the complexity of preprocessing, I did not include this part

2) *Cache*: Using a DynamoDB table to store the record is a good choice. However, if we just need a cache, Redis will be a better choice actually. Because it's cheap and designed for cache. Redis could be used with AWS ElastiCache

3) *Scalability*: What if there are millions of users or movies? Although S3 could hold them, but it will be hard when we want to retrieve some information of a specific user. I think HBase may be a better choice as a NoSQL database

4) *Echo*: Finally, maybe it's better to realize more functions and more dialogues in Alexa

REFERENCES

- [1] "Facebook, Pandora Lead Rise of Recommendation Engines - TIME". TIME.com. 27 May 2010. Retrieved 1 June 2015.
- [2] https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223
- [3] Sallab, Ahmad EL, et al. "Deep reinforcement learning framework for autonomous driving." Electronic Imaging 2017.19 (2017): 70-76.
- [4] Zhang, Ben, et al. "The Cloud is Not Enough: Saving IoT from the Cloud." HotCloud. 2015.
- [5] <http://flask.pocoo.org/>
- [6] <https://getbootstrap.com/>
- [7] DeCandia, Giuseppe, et al. "Dynamo: amazon's highly available key-value store." ACM SIGOPS Operating Systems Review. Vol. 41. No. 6. ACM, 2007
- [8] <https://docs.python-guide.org/dev/virtualenvs/>
- [9] <https://github.com/benoitc/gunicorn/releases>
- [10] Murenin, Constantine A. (18 February 2013). "A dynamic web-site written wholly in nginx.conf? Introducing mdoc.su!". nginx@nginx.org (Mailing list). Retrieved 24 December 2014
- [11] Hendrickson, Scott, et al. "Serverless computation with openlambda." Elastic60 (2016): 80.