The University of Hong Kong

Faculty of Engineering

Department of Computer Science

COMP7705

Project Report

Real-time E-Commerce Recommender System Based on Apache Flink

Submitted in partial fulfillment of the requirements for the admission to the degree of Master of Science in Computer Science

By

Wu Bijia (3035561649) Zhu Shengda (3035562411)

Supervisor: T.W.Chim
Date of submission: 31/7/2019

Content

1. Introduction

This chapter mainly introduces the research background and significance of the real-time e-commerce recommender system. It then clarifies the work and goals to be completed in this project and summarizes the basic content together with the organizational structure of this article in the end.

1.1 Background

In recent years, the rapid development of information technology has made the Internet more popular and applicable. But it also brought the problem of information overload. In the past, people only got information from traditional media such as newspapers and made corresponding decisions. The scarcity and asymmetry of information has become a key factor affecting people's decision-making. But now, the sheer volume of information makes it difficult for people to find content that is of real interest; relatively, valuable information makes it harder for them to stand out in the vast amount of information.

In order to effectively solve the problem of information overload, two solutions have been proposed. One is a search engine[1] and the other is a recommender system. For search engines, specific keywords can quickly locate the information users want to find, which can also be called explicit search. For example, the search engine system represented by Google and Baidu provides powerful help for users to retrieve information. But search engines are not omnipotent. The same search keywords will return the same content, which is contrary to the individualized and diversified requirements of people in the new era. For example, when consumers are shopping on an e-commerce platform, everyone has their own special preferences. It is obviously unreasonable

to recommend the product to users only through keywords. For a specific product, how to recommend the product to a specific group is also a Big problem. At this time, you need to recommend the system.

The recommender system, also known as implicit search, will play an irreplaceable role in e-commerce and other scenarios.

Recommender systems(RSs) are being used by an ever-increasing number of E-commerce sites to help consumers find products to purchase. For example, Amazon[2]-[4], eBay[5], Taobao[6] and Jingdong[7] all have their own specialized RSs. According to VentureBeat statistics, RS has brought in 20% - 30% more sales for Amazon[8] which reveals the value of RSs.

In the field of E-commerce RSs, the basic algorithms include collaborative filtering algorithms [9][10] and content-based algorithm[11]. However, these algorithms still have some shortcomings. For example, when a user generates some new log files(such as adding an item into shopping cart, adding an item to favorites), the RS's recommending list generally won't change immediately until some delay while the user may already have closed the website. Besides the performance in responsiveness, another big issue is the precision of the RS. Some more advanced algorithms are then proposed in order to address this issue. Youtube[12] uses deep neural network algorithm for its own RS. Taobao[13] applies deep interest network algorithm to its RS. Besides, in many articles[11]−[13], they also raise their own RS architectures oriented to E-commerce mass data in the cloud.

However, algorithms and the architecture in E-commerce RSs are still limited for an increasing demand for higher precision, higher CTR (click-through rate) and faster responsiveness.

1.2 Subject

In this recommender system, we will build a complete, real-time e-commerce recommender system based on the Flink platform and combine various recommendation strategies. The main content is:

First, to build the overall framewor, including the construction of Hadoop cloud, message queue and database selection.

Second, to combine with multiple recommendation strategies, create offline and online models to provide generalization and personalized recommendations.

Third, to build a simple web UI module and evaluate the overall model.

1.3 The Organization of Report

Chapter 1 introduces the background and significance of recommendation system and the research status of recommendation system is summarized.

Chapter 2 introduces the key theory and technology of recommendation system. The process of recall and ranking in recommendation and the basic algorithm are introduced in detail.

Chapter 3 introduces the deployment process of cloud cluster, the framework and technology used, the operation and application of related databases according to the project requirements.

Chapter 4 introduces the data set, model selection, model training and corresponding metric of the project, and also includes the comparison between different algorithms.

Chapter 5 summarizes the contents and shortcomings of the whole project, as well as the outlook after R&D.

2. Theoretical Analysis

This chapter focuses on the key theories and techniques related to recommender systems. Firstly, this chapter elaborates the concept of recommendation system and its input and output. Then the typical recommendation system algorithms and features are introduced, which lays a foundation for the construction of subsequent recommendation systems and the use of recommendation algorithms.

2.1 Recommender System

The recommender system provides an effective solution to solve the problem of information overload and users' demand for the recommendation system has penetrated into all aspects of life. This section mainly describes the concept of the recommendation system and introduces the input and output of the recommendation system.

2.1.1 Concept of Recommender System

The recommendation system serves as an information filtering system which is widely used to provide users with personalized recommendations. Recommended objects include news, music, movies, e-commercial products, books, etc.

There are currently three ways to solve the information overload problem, including catalogs, search engines and recommendation engines. That is to say, Yahoo, Hao123 as the representative of the category of intelligent coverage of popular classification. Google, Baidu as the representative of the search engine to help users search with keywords. However, in order to meet the requirements of individualization and diversification of users in the new era, the

recommendation system is a future trend. The so-called recommendation system, through studying of user preferences and user information, models the user and models the recommended objects. Based on the relevant recommendation algorithm, the recommendation system pushes the content that the user may be interested. A general model of recommendation system is shown below.

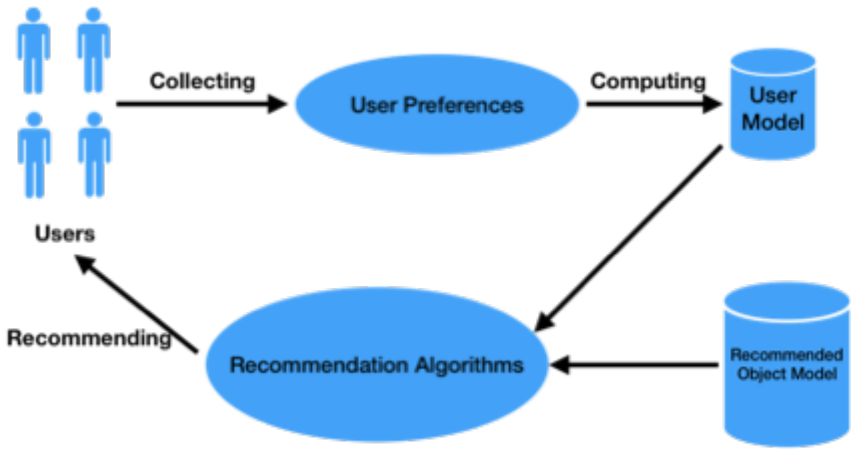


Figure 1: Recommendation System General Model

The recommended system paradigms can be mainly divided into 3 types: completely non-personalized paradigm, associated recommendation paradigm, fully personalized paradigm. Completely non-personalized recommendations include popular recommendations, such as various leaderboards. Associated recommendations include correlations between associated user recommendations and associated items. Fully personalized recommendations mean specific content for specific users. This is one of the most attractive features of the recommendation system.

A good recommendation system not only provides users with accurate recommendation services, but also enhances the user experience and retains users. It should also solve some typical problems of recommendation systems such as long tail effect and cold start.

2.1.2 Input & Output of Recommender System

In general, the input of the recommendation system includes items, users and review. The features of items are different in different fields. For example, when recommending e-commercial products, the features may include product category, price, color, brand, etc. When recommending news, item features may include title, body, picture, author, time, etc. That's why there's need building item models.

Similarly, the user model is different in different scenarios. For example, when recommending e-commercial products, user features may include gender, income, city of residence, etc. When recommending news, user features may include user tags, city residence, page staying time, etc. It's the reason for building a user model.

Review is the bridge between the user and the item. Common reviews include rating, browsing behaviors, buying behaviors, etc. Therefore, in different fields, the review is also different. Reviews can be explicit, such as ratings, comments, etc. Reviews can also be implicit, such as browsing time.

The recommendation system uses the recommendation algorithms to combine the user, item, and review features to generate output which is a series of recommended items. At the same time, the recommendation system will sort these recommended items in order to enhance the user experience and retain users.

2.2 Recommendation Algorithm

The core of recommendation system is to select suitable products from a large number of commodity libraries and finally display them to users. Because of the huge number of commodity stores, the common recommendation system is generally divided into two stages, namely, recall stage and ranking stage. In the recall stage, a small number of candidate sets that users may be interested in are obtained from the full merchandise library. In the ranking stage, the candidate sets obtained in the recall stage are sorted accurately and recommended to users.

2.2.1 Candidates Generation (Recall)

2.2.1.1 Introduction

The recall stage can be regarded as selecting a small set of candidates from a large number of items according to the historical behavior data of users, which is equivalent to rough sorting. On the one hand, this process can provide personalized recommendation for users; on the other hand, it can also reduce the amount of computation for the subsequent ranking process.

At present, the recommendation system in industry generally adopts multi-channel recall strategy in the recall stage. In a personalized recommendation, we usually need to use a variety of strategies at the same time. If you try to recommend only through a refined recommendation strategy, users tend to be very interested in the initial stage, and as the number increases, users will gradually become tired. After all, users tend to read in a pluralistic way, especially in the field of e-commerce. If they always recommend the same kind of goods that they have clicked on or purchased, the enthusiasm of users to buy will decrease sooner or later. This also does not use e-commerce platform to expand the user's purchasing ideas.

The input of the recall algorithm is varied, including not only the user's personalized portrait, but also the distribution of platform traffic. The output of the recall is a subset of the merchandise and is exported to the ranking process.

2.2.1.2 Recall Algorithm

Generally, recall strategies do not require very precise results, but use different strategies to recall items that users may be interested in from different perspectives to expand the coverage of recommended items. So recall algorithms are usually not particularly advanced algorithms, such as collaborative filtering, or just some statistical rules.

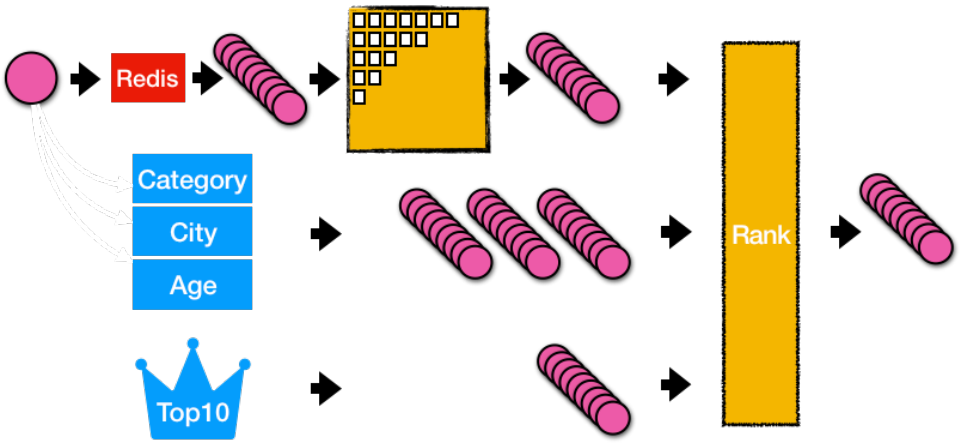


Figure 2: multi-channel recall strategy

As shown above, three strategies are used to recall goods, each of which generates a relatively large set. Then let’s introduce some common recall algorithm used in the project.

1) Collaborative Filtering

Collaborative filtering mainly obtains explicit or implicit information about items through user's historical behavior. That is to say, according to the user's preference for items, to discover the

correlation between the user and the item itself or between them, and then recommend the user according to the correlation. This correlation is usually represented by a sparse matrix.

Collaborative filtering is divided into user-based collaborative filtering and item-based collaborative filtering.

Shortly speaking, the idea of user-based collaborative filtering is that user A may like products that user B likes with similar hobbies. So this algorithm needs to maintain a similarity matrix between users and users. When recommending products to users, first of all, we need to find one or more users who are most similar to the user and find the products they are interested in. These products are considered to be products that the user may like. However, for the current e-commerce platform with large user base, it is difficult to maintain such a large matrix, which not only consumes a lot of memory, but also takes a long time to calculate the closest users. Secondly, cold start problem, that is, recommendation accuracy for users with no or few history. However, because the recall process does not require high accuracy, it is also necessary.

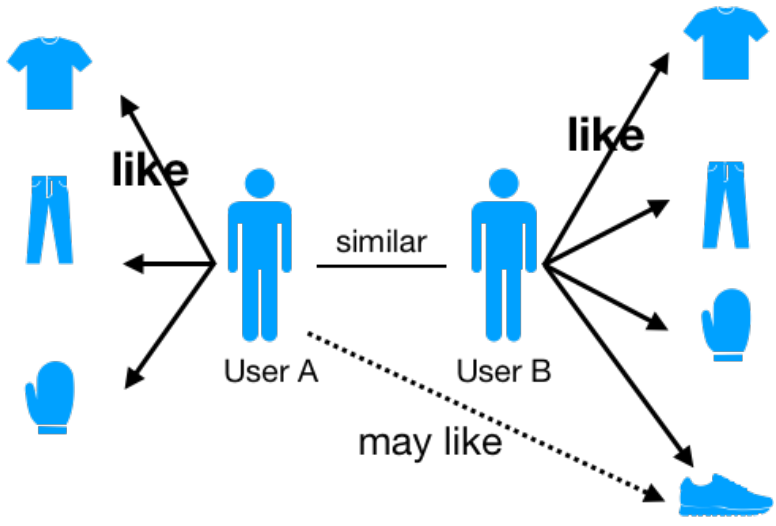


Figure 3: User-based CF

On the other hand, the idea of collaborative filtering based on items is that users may like items similar to those they used to like. It differs from previous algorithms in that it calculates similar items rather than similar users. In other words, item-based collaborative filtering needs to maintain an item similarity matrix. In fact, for the self-operated e-commerce like Amazon, the dimension of the item matrix is much smaller than that of the user matrix, and the amount of calculation is much lower.

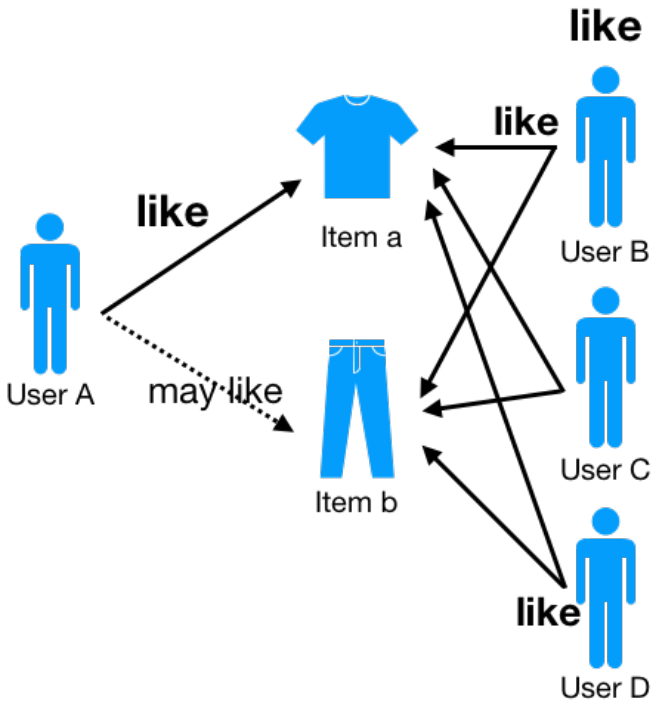


Figure 4: Item-based CF

The maintenance of this matrix is shown in the figure below. The similarity between each item and other items is expressed by the formula shown in the picture. Numerator indicate the intersection of users who like item a and item b, and denominators indicate the union of users who like item a and item b. If it is 1, it means that the two goods are completely similar. If it is 0, it

means that the users who like the two goods intersect at 0, it also means that the two goods are completely different.

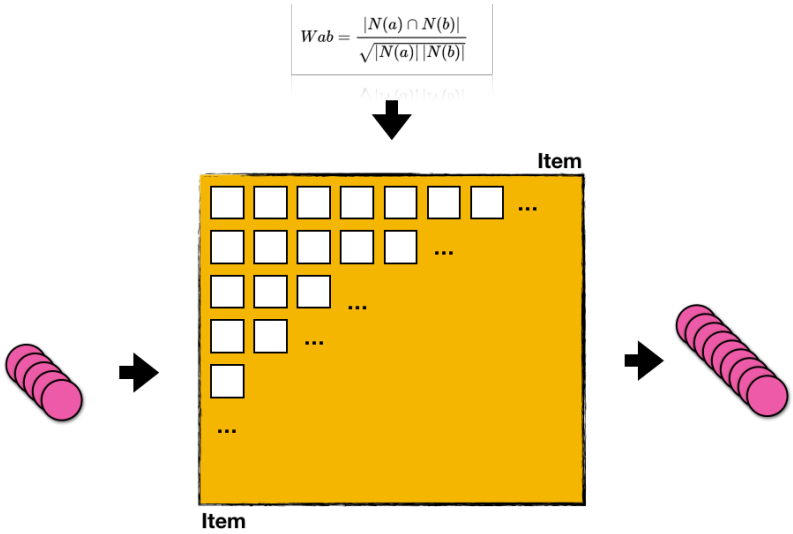


Figure 5: Item similarity matrix

According to the matrix of the above graph, each item can be regarded as a column vector. When the similarity between an item and other items is required, only the similarity between the column vector of this item and other vectors can be required. This similarity can be expressed by cosine.

2）Profile-based Recall

Generally speaking, the input of recommendation system should also include user profile, that is to say, the label entered when the user registers. The characteristics of users are generally gender, age, city, province, etc. In the offline model, we will make statistics on the purchase habits of users of different gender, age, city and so on. For example, teenagers aged 12-18 are definitely different from parents over 30 in buying things. So when a user's new log arrives, we first extract the information in his/her profile, and then for different domains, we recall items that may be of interest to him/her in this domain.

3）Content-based Recall

Content-based recalls are mainly based on content portraits obtained by previous NLP. The recalls are established by the weight of item corresponding to classification/subject/keyword. The recalls are sorted according to the corresponding weight of user portraits and the distance of content portraits.

2.2.2 Ranking Algorithm

2.2.2.1 Introduction

The input of the ranking algorithm is a subset of items returned in the recall phase, and the output is a candidate set recommended by the end user. Because in the actual business, we will take into account repetitive recommendation, pornographic filtering and other issues, so in the final recommendation to users of goods, but also through weight removal, manual rule filtering, etc., here no longer continue.

2.2.2.2 Ranking Algorithm

In fact, there is no clear boundary between the recall algorithm and the ranking algorithm. Recalls tend to increase the recall rate, while rankings tend to improve the accuracy. Therefore, the advantages of different models are reflected here. The model of sorting algorithm is usually more complex and computational requirements are higher. In this paper, only a few widely used and recognized ranking algorithms are introduced.

1) Logistic Regression

Logical regression is a generalized linear regression model, which is also the elder of machine learning task.

Take a user's user portrait vector and stitch up the object portrait vector of an item. If the user has ever been associated with the item, label is 1. These can be regarded as positive samples. At the same time, the user can "skip" item as negative samples, label is 0. Fitting the following equation:

$$y = \frac{1}{1 + e^{-(w^T x + w_0)}}$$

In the figure above, x is a log vector, W is a training parameter, and the loss function is：

$$loss = \sum_{(x,y)\in D} -y \log(y') - (1-y) \log(1-y')$$

Where y is the label 0 or 1 of the sample, y' is the number between 0 and 1 predicted by the model.

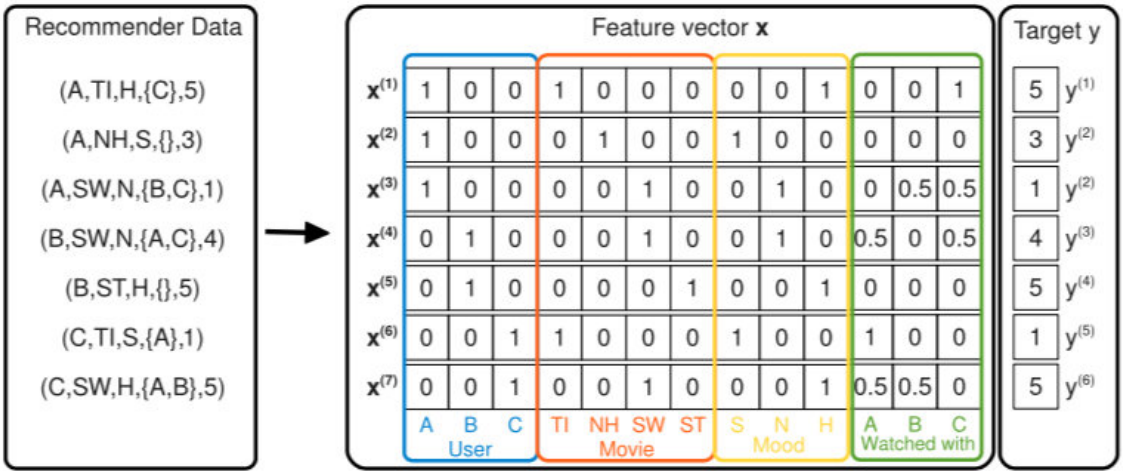
Logic regression has the advantages of simple model, good bug repair operation and explanatory ability. Each dimension of the model can be seen intuitively. At the same time, because the model is simple, the amount of calculation is not large, the resource consumption is small, and the speed is fast.

On the other hand, his disadvantage is that the ceiling is easy to reach. Because he did not consider the cross-feature, he could only choose the feature manually. If he wanted to introduce the non-linear factor, he needed to do the cross-feature, which could easily produce billions of features.

2) Factorization Machines (FM)

FM introduces second-order feature combination on the basis of logistic regression[14], and does not equip every second-order cross-feature with a super-parameter, because this will reduce the generalization ability of the model (some second-order items may not be able to train their parameters because there has been no all of them in the training set, especially in large-scale sparse. This problem is particularly prominent in scenarios where sparse features exist.

The following figure is an intuitive representation of FM model：



Intuitive Interpretation of FM

FM introduces a word called context. Context is not an intuitive description of item or user, but a dynamic feature that users have at the moment of clicking or scoring, such as mood, time, etc.

For contextual recommendation, the previous method, Tucker decomposition, or Multiverse Recommendation, is mentioned in the paper. The main idea is to transform an M-mode tensor into a smaller core tensor and to match each model with a one factor matrix V. The disadvantage of this model is that Time complexity is too high.

The generative formula of FM model can be expressed as:

$$\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$$

FM model also directly introduces the second-order feature combination of any two features. The biggest difference between SVM model and FM model is the method of calculating the weight of feature combination. For each feature, FM learns a one-dimensional vector of size k, so the weight of the combination of two features xi and xj is expressed by the inner product < vi, vj > of the corresponding vectors vi and vj. This is essentially embedding characterization of features.

Embedding here does not depend on whether a particular feature combination has ever appeared. So as long as the combination of feature xi and other arbitrary features has appeared, then we can learn our embedding vector. This is also the reason why the model has strong generalization ability.

3) Field-aware Factorization Machines (FFM)

The full name of FFM is Field-aware FM[15], which can be translated intuitively into FM model that can realize the existence of field.

		Publisher	Advertiser
+80	−20	ESPN	Nike
+10	−90	ESPN	Gucci
+0	−1	ESPN	Adidas
+15	−85	Vogue	Nike
+90	−10	Vogue	Gucci
+10	−90	Vogue	Adidas
+85	−15	NBC	Nike
+0	−0	NBC	Gucci
+90	−10	NBC	Adidas

Artificial CTR data set

FM can be called a special case of FFM. FFM adds a field concept when considering data sets. As shown in the chart above, Publisher and Advertiser represent two fields. In Publisher field, ESPN, Vogue and NBC represent features in this field. In FFM, features are still represented by embedding vectors. When considering crossover features, each feature uses different embedding when crossing features in different fields.

This means that if there are F feature domains, then each feature is expanded from a k-dimensional feature embedding of FM model to (F-1) k-dimensional feature embedding. It's F-1, not F, because features don't fit in with you, so you don't have to think about itself.

FFM is slightly bulky in the actual operation process, which is compared with FM model. We can deduce that if the model has n features, then the parameter of FM model is n*k, where k is the size of the eigenvector. What about the parameters of FFM model? Because each feature has (F-1) k-dimensional feature vectors, its model parameters are (F-1)*n*k, that is to say, the parameters are more than (F-1) times larger than that of FM model. This means that if our task has 100 feature domains, the parameters of FFM model are about 100 times larger than that of FM model. This is actually very scary, because in real tasks, the number of features n is a very large number, and tens of hundreds of feature fields are also common.

In addition, the FM model can be rewritten by formula to reduce the computational complexity of the square originally seen as n to O(n*k). FFM is not able to do similar rewriting, so its computational complexity is O(n^2 * k), which is obviously much slower than FM model in computational speed.

Because the parameters of FFM model are too large, it is very easy to over-fit when training FFM model, so it is necessary to take early stop and other means to prevent over-fitting. According to experience, the K value of FFM model can be smaller.

3. System Architecture

This chapter focuses on our design of real-time e-commerce recommendation system. Including the overall network structure and system architecture design, functional module design and system database design. The selection and design ideas of various software in the design process lay the foundation for the development of a recommendation system with high stability, fast response and superior performance.

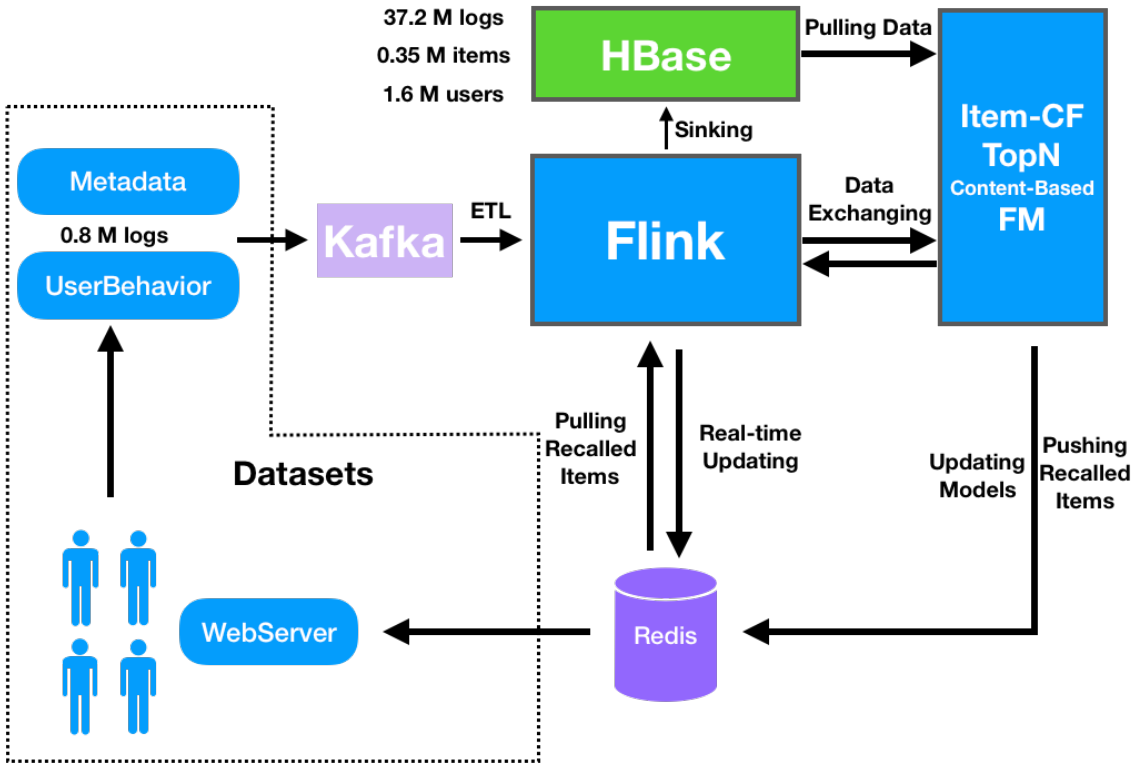
3.1 Network Architecture

The system designed in this paper is based on C/S architecture, and the system network structure diagram is shown in the figure below. In this system, the user browses the product through the web interface, that is, uses the recommendation system. Of course, the recommendation system is basically transparent to the user, and the user only needs to enjoy the recommendation result. The network transmits the interaction behaviors of users between items, such as browsing and purchasing, to the server to provide data for the recommendation engine. The recommendation engine will then use the logs for recommendation result calculations and data storage.

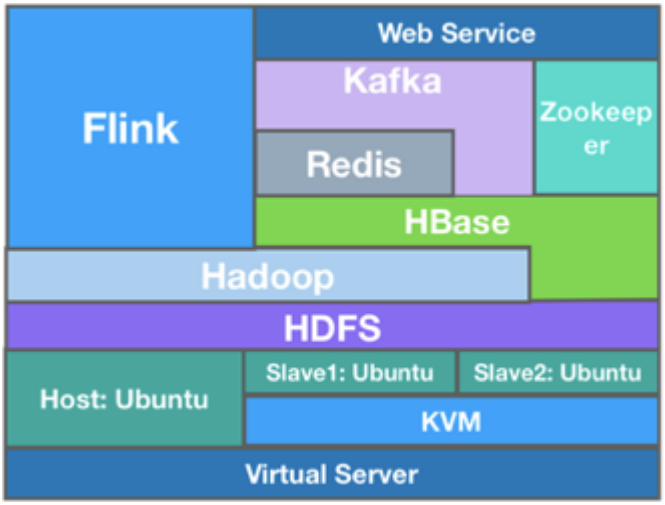
System Network Architecture

3.1.1 Architecture Overview

Our system architecture is shown below. The web server acts as the user's interface, and the user obtains the recommendation result and provides real-time data flow for the recommendation system. The data stream is distributed to the Flink engine through the kafka message queue. Flink performs model establishment and real-time recommendation result calculation, and stores data and recommendation results in online database Redis and offline distributed database HBase. The underlying machine simulates the establishment of a cloud environment on a virtual server.



System Workflow

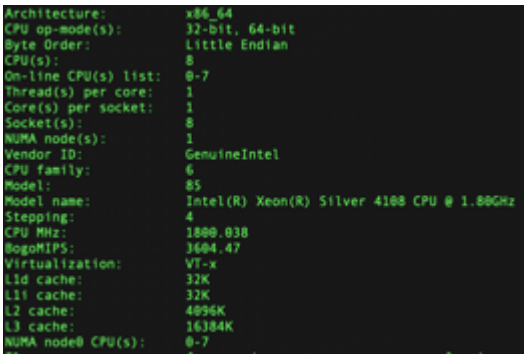


System Architecture

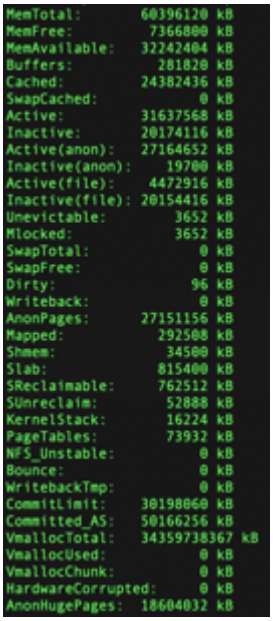
3.1.2 Host & Slaves Parameters

Our underlying machine is a virtual server that we apply to the college. In order to simulate a distributed environment, we need to use virtualization technology to create two virtual machines with linux-based os. Here we use KVM(kernel-based virtual machine), which allows us to create virtual machines in a server environment (non-desktop environment). This is the main difference between KVM and Virtual box, Xen and other virtualization technologies the reason we choose KVM.

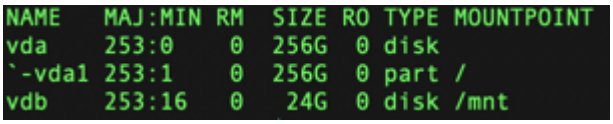
The overall memory and cpu information are as follows:



CPU Information

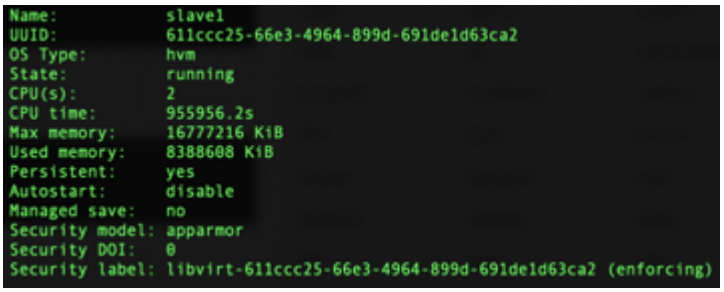


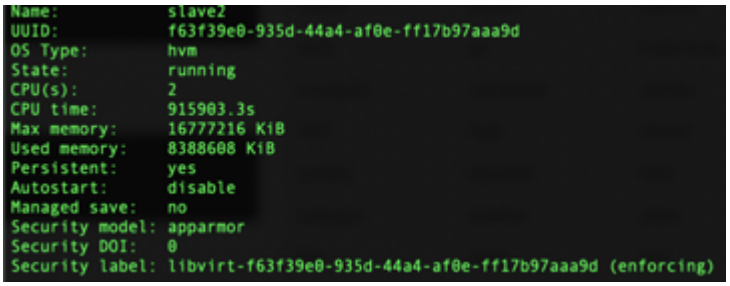
Memory Information



Disk Information

In order to meet the needs of both distributed storage and distributed computing, our resource allocation for slave1 and slave2 is as follows:





3.1.3 Hadoop Cloud

In today's industrial level recommendation systems, the rss are deployed in the cloud to meet the requirements of high availability, high stability and high scalability. In order to simulate the real recommendation system environment, we built a cloud environment where the recommendation models are established and recommendation algorithms are applied. Due to the limitations of hardware conditions, our “cloud” is rather small while the recommendation system built on the cloud can be easily transferred to the industrial environment, which reflects the high portability of the recommended system we built.

3.2 Functional Module Design

3.2.1 Central Computing Module: Flink

With the advent of the artificial intelligence era and the burst of data volume, the most common practice of data services in a typical big data business scenario is to use batch processing technology to process full amount of data and stream computing to process real-time incremental data. In most business scenarios, the user's business demand is often the same in batch and stream processing. However, the two sets of calculation engines that users use for batch processing and stream processing are different.

Therefore, users usually need to write two sets of code. There is no doubt that this brings some extra burden and cost. Can we use a unified set of big data engine technology, so that users only need to develop a set of code according to their own business demand? In this way, in a variety of different scenarios, whether it is full or incremental data, or real-time processing, a set of solutions can be fully supported, which is the background and original intention of Flink.

At present, open source big data computing engines have many choices, such as Storm, Samza, Flink, Kafka Stream, etc.; batch processing such as Spark, Hive, Pig, Flink, etc. There are only two options for computing engines that support both stream processing and batch processing: one is Apache Spark and the other is Apache Flink.

From the comprehensive consideration of technology, ecology and other aspects. First of all, Spark's technical philosophy is based on the calculation of batch flow. Flink, on the other hand, uses a stream flow to simulate batch calculations. From the perspective of technology development, there are certain technical limitations in using batch to simulate streaming, and this limitation may be difficult to break through. While Flink simulates batches based on streams, it is technically

more scalable. Therefore, more and more companies now use Flink as the recommendation engine. For example, Alibaba has researched Blink as their recommendation engine based on Flink. That's one of the reasons why we chose Flink instead of Spark.

Flink is a low-latency, high-throughput, unified big data computing engine. At the same time, the Flink computing platform runs on top of the open source Hadoop cluster. YARN using Hadoop is used as resource management scheduling, and HDFS is used as data storage. Therefore, Flink can seamlessly interface with the open source big data software Hadoop. At the same time, Flink is the most different from other stream computing engines. In fact, it is state management. What is the state? For example, if you develop a system or task for stream computing, you may need to perform statistics on the data, such as Sum, Count, Min, and Max. These values need to be stored. Because of the constant update, these values or variables can be understood as a state. If the data source is reading Kafka, RocketMQ, you may want to record where to read, and record Offset, these Offset variables are the state to be calculated.

Flink provides built-in state management that can be stored inside Flink without having to store it in an external system. The advantage of this is that the first is to reduce the computing engine's dependence on the external system and deployment, so that the operation and maintenance is simpler. Second, the performance is greatly improved: if you access it through external, such as Redis, HBase it must It is through the network and RPC. If accessed through Flink internally, it only accesses these variables through its own process. At the same time, Flink will periodically check these states for Checkpoint and store Checkpoint in a distributed persistence system, such as HDFS. In this case, when there is any failure of Flink's task, it will restore the state of the entire stream from the most recent Checkpoint, and then continue to run its stream processing. There is no data impact on the user.

In our architecture, we employ Flink's various advantages combined with kafka, Redis and HBase building a recommendation system.

3.2.2 Data Access Module

Message Queue Middleware (referred to as message middleware) refers to the use of efficient and reliable messaging mechanism for platform-independent data exchange, which integrates distributed system based on data communication. By providing a message passing and message queuing model, it can provide application decoupling, elastic scaling, redundant storage, traffic peaking, asynchronous communication, data synchronization, etc. It plays an important role as in a distributed environment. Currently open source messaging middleware includes ActiveMQ, RabbitMQ, Kafka, RocketMQ, ZeroMQ and so on.

Among the many message middlewares, we chose Kafka. Kafka was originally a distributed, multi-partition, multi-copy, zookeeper-based distributed messaging system developed by LinkedIn using the Scala language. It is a high-throughput distributed publish-subscribe messaging system that is widely used for horizontal scalability and high throughput. More and more open source distributed processing systems such as Cloudera, Apache Storm, Spark, Flink, etc. support integration with Kafka.

In the recommendation system, Kakfa's characteristic of high throughput, automatic persistence, automatic load balancing and processing of streaming data makes Kafka the preferred messaging middleware for user log processing. This is why we chose Kafka. In our architecture, user logs are stored in the Kakfa queue, and Flink acts as a consumer to fetch log data directly from Kafka's topic and process the data. The processed data is persisted to HBase, and the processed data is used to calculate the recommended result which finally is stored in Redis. The user then interacts with

the recommendation results in Redis through the web interface to form new log information and send it to Kafka. Such a complete closed loop recommendation system is formed then.

3.2.3 Offline Model

The offline model is scheduled by Flink. According to the timestamp in the data set, the offline model is triggered once every two hours according to the event time. The model process first takes the full user log from HBase to the first two months, and then carries out all the L offline processes. The whole offline model will run a little longer. During the whole process, the data in Redis will be updated accordingly for the next online model.

The offline model steps are as follows:

1. Take the full user log from HBase from the current to the first two months, and generate the total user set and item set so far to determine whether the user is a new user after that and whether it needs cold start. This process stores collections in memory.

2. Join log and metadata, establish positive and negative samples, and use them as input of sorting model. In this process, you need to optimize the entire data, because the device has limited memory. The optimization method is to optimize the data type according to the number of feature types. Similarly, because of the limited memory, we only selected nine features ['age','sex','city','county','brand','shop_id','cate','user_id','sku_id'] as the input of the model.

3. Conversion format. Since FM input requires a specific libffm format, it is necessary to convert the DataFrame into libffm format. After the transformation, the corresponding category memo needs to be saved and used as the new log for the corresponding mapping transformation. This memo is stored in memory for easy real-time invocation.

```
hduser@msp18033s1:~/Ecom-Recommend-Algo/data$ head train_ffm_model.txt
1 0:5:1 1:246:1 2:5341:1 3:18341:1 4:21891:1 5:21952:115291.0 6:21953:9.789564 7:473885:1 8:951832:1
1 0:3:1 1:358:1 2:2856:1 3:15559:1 4:21896:1 5:21952:24976.0 6:21953:9.393827 7:395877:1 8:1026883:1
1 0:5:1 1:47:1 2:11486:1 3:15828:1 4:21888:1 5:21952:17540.0 6:21953:9.370503 7:139828:1 8:942433:1
1 0:4:1 1:336:1 2:3563:1 3:13339:1 4:21942:1 5:21952:0.0 6:21953:0.0 7:82302:1 8:823807:1
1 0:1:1 1:353:1 2:1514:1 3:14760:1 4:21876:1 5:21952:0.0 6:21953:0.0 7:138206:1 8:841868:1
1 0:3:1 1:255:1 2:2856:1 3:15717:1 4:21910:1 5:21952:0.0 6:21953:0.0 7:648269:1 8:1082138:1
1 0:4:1 1:317:1 2:5105:1 3:18038:1 4:21936:1 5:21952:2133.0 6:21953:9.797265 7:523752:1 8:1036606:1
1 0:4:1 1:317:1 2:5105:1 3:18038:1 4:21936:1 5:21952:2133.0 6:21953:9.797265 7:523752:1 8:911141:1
1 0:4:1 1:306:1 2:6221:1 3:18990:1 4:21876:1 5:21952:29941.0 6:21953:9.436308 7:275401:1 8:912970:1
1 0:4:1 1:209:1 2:10292:1 3:12583:1 4:21891:1 5:21952:22835.0 6:21953:9.490836999999999 7:572219:1 8:874101:1
```

4. Adding up is the recall process. Our system mainly adopts three recall methods: item-item CF, calculates the corresponding vector of each item, and then puts 20 items that each user likes best into Redis. (Another is to put the vector corresponding to item into Redis and calculate the user's favorite items in real time when the user comes). Content-based: We selected city and age as filtering features, and put K items that users of different cities and ages like into Redis. Hottest: Return the hottest merchandise collection in different periods (1 month, 1 week, 1 day) to y) to Redis.

5. FM model training. The parameters of the model are saved in Redis after training. As shown below, set the parameter name to key and save the parameter value in Redis.

```
bias: -1.30287
i_0: -0.180989
i_1: 0.122972
i_2: 0.096754
i_3: -0.0352964
i_4: -0.0947834
i_5: -0.0509059
```

3.2.4 Personalized Model (Online model)

The parameters of FM model:

```
param = {'task':'binary',
         'lr':0.2,
         'lambda':0.002,
         'metric':'auc',
         'opt': 'adagrad',
         'k': 16
        }
```

3.2.4 Personalized Model (Online model)

When a user log is transmitted from Flink via TCP socket, a new thread will start, and the thread will perform the following process:

1. Return items that user clicked 10 times through Redis. When Flink accepts the new log, it not only passes the log into the HBase full log, but also passes the log into the EDIS with the user as the key, and the data structure is list.

2. Get the city and age of the user and return the items candidates of the top N corresponding to the city and age.

3. Get the current hottest item candidates

4. Through 1, we get items candidates after collaborative filtering. According to the item similarity matrix of collaborative filtering, 10 latest items in 1 are added with each item's similar vectors according to the time attenuation weight. Finally, the final user item vector is obtained, and the top item with the largest value in the vector is returned as candidates.

5. Integrate items in 2, 3, 4.

6. Using the previous category memo, the corresponding items2ffm vectors are generated.

7. Pass the vector through Kafka to Flink. Because of the large amount of data in libffm format and the same format in each line, Kafka is not suitable for socket transmission, but for Kafka transmission.

3.3 Database Design

3.3.1 Database Comparison Analysis

1. Why use NoSQL instead of a relational database management system like MySql?

The traditional relational DBMS has good performance, high stability, history, is simple to use, powerful, and has accumulated a large number of successful cases. In the Internet world, MySQL has become one of the most popular relational databases. In the 1990s, a website did not have a large amount of traffic, and it was easy to cope with a single database. At that time, more were static web pages, and there were not many websites with dynamic interaction types. However, with the rapid development of the website, the number of visits has risen sharply. Most websites that use the MySQL architecture have begun to have performance problems with the DBMS. The web application is no longer only focused on functions, but also in pursuit of performance. Programmers have begun to use a lot of caching techniques to ease the pressure on the database and optimize the structure and index of the database. It is popular to use file caching to alleviate database pressure, but when the number of visits continues to increase, multiple web machines cannot be shared through the file cache, and a large number of small file caches also bring higher IO pressure. At this time, Memcached naturally became a very fashionable technology product.

As a stand-alone distributed cache server, Memcached provides a shared high-performance cache service for multiple web servers. On the Memcached server, it has developed an extension of multiple Memcached cache services based on the hash algorithm. Consistent hashes address the drawbacks of increasing or decreasing the cache servers causing a large amount of cache invalidation caused by re-hashing.

Due to the increased write pressure on the database, Memcached can only alleviate the reading pressure of the database. Reading and writing are concentrated on one database to make the database overwhelmed. Most websites start to use master-slave replication technology to achieve read-write separation to improve read and write performance and read library scalability. Mysql's master-slave mode became standard of website of that time.

With the continued rapid development of web2.0, on the basis of Memcached's cache, MySQL's master-slave replication and read-write separation, the write pressure of the MySQL main library begins to be the bottleneck. And the amount of data continues to soar. Because MyISAM uses table locks, severe lock problems can occur under high concurrency. And large number of high-concurrency MySQL applications start to use the InnoDB engine instead of MyISAM. At the same time, it has become popular to use sub-tables to alleviate the problem of expansion of write pressure and data growth. At this time, sub-databases became a hot technology. MySQL introduced a table partition that is not yet stable. Although MySQL has launched the MySQL Cluster cluster, there are few successful cases on the Internet. The performance cannot meet the requirements of the Internet, while it provides a very large guarantee for high reliability.

On the Internet, most of MySQL should be IO-intensive, and MySQL application development in high-volume and high-concurrency environments is becoming increasingly complex and technically challenging. The rules of the sub-database are all required to be experienced. Although a powerful company like Taobao has developed a transparent middleware layer to shield developers from the complexity, it can't avoid the complexity of the entire architecture. Sub-libraries of sub-database sub-tables face expansion problems at a certain stage. There is also a change in demand, which may require a new way of sub-library.

MySQL databases also often store large text fields, which makes the database tables very large. When doing database recovery, it is very slow, and it is not easy to quickly recover the database. For example, 10 million 4KB of text is close to 40GB. If you can save this data from MySQL, MySQL will become very small.

The relational database is very powerful, but it does not work well for all application scenarios. MySQL's poor scalability (requires complex technology to implement), IO pressure under big data, table structure changes are difficult.

The emergence of NoSQL database makes up for the shortcomings of relational data (such as MySQL) in some aspects, which can greatly save development and maintenance costs in some aspects. The advantages of NoSQL include:

1).Easy to expand

There are many different types of NoSQL databases, but one common feature is to remove the relational nature of relational databases. There is no relationship between the data, so it is very easy to expand. Also intangible, it brings scalable capabilities at the architectural level.

2).Large amount of data, high performance

NoSQL databases have very high read and write performance, especially in the case of large data volumes. This is due to its non-relationship and the simple structure of the database. Generally, MySQL uses Query Cache. Each time the table is updated, Cache is invalid. It is a large-grained Cache. Cache performance is not high in applications with frequent interactions with web2.0. NoSQL's Cache is record-level and is a fine-grained Cache, so NoSQL is much more powerful at this level.

3).Flexible data model

NoSQL can store custom data formats at any time without having to create fields for the data to be stored. In a relational database, adding or deleting fields is a very cumbersome task. If it is a very large amount of data, adding a field is simply a nightmare. This is especially true in the web2.0 era of large data volumes.

4).High availability

NoSQL can easily implement highly available architectures without compromising performance. For example, Cassandra, the HBase model, can also be highly available by copying the model.

In the recommendation system, the system needs to be highly available, highly scalable, and extremely demanding on data IO performance. Therefore, NoSQL is a better choice than traditional relational databases.

2.Why use redis？

As mentioned in the previous section, the actual MySQL is suitable for massive data storage by loading hot data into the cache through Memcached to speed up access time. Many companies have used such an architecture. However, as the amount of business data continues to increase and the number of visits continues to grow, many problems will be encountered. Thus more companies choose NoSQL as their DBMS.

In recent years, there have been many kinds of NoSQL products in the industry. So how to select the NoSQL products suitable for our recommendation system, how to use these products correctly and to maximize their strengths should be thought deeply. The truth is to understand the

positioning of these products, and to understand the tradeoffs of each product aiming to choose the right product and to achieve strengths and avoid weaknesses.

In general, the NoSQL products we choose should have the ability to solve the following problems:

1). A small amount of data storage, high-speed read and write access. In the recommendation system, we need to process a large number of small text user logs in real time. These processed logs need to be quickly stored in the database for online model calculation. Therefore, the products we choose need to ensure high-speed access through the data in-momery providing the function of data landing. This is actually the main application scenario of Redis.

2). Massive data storage, distributed system support, data consistency guarantee, and convenient cluster node addition/deletion. In the recommendation system, the amount of users and the amount of logs are very large, which means that the traditional relational DBMS cannot meet our needs and we need NoSQL products that support distributed storage.

3). Schema free, auto-sharding, etc. In order to maintain the flexibility and portability of the recommendation system, the NoSQL product we choose should also have the ability to adjust the schema at any time.

After the above analysis, Redis is best suited for data in-momory scenarios. Redis is an in-memory NoSQL DBMS that is often used as a database, cache, and message broker. Unlike other in-memory DBMS, it also saves data to disk and also provides a number of data structure types (Sets, Sorted Sets, Hashes, Lists, Strings, Bit Arrays, HyperLogLogs, and Geospatial Indexes). Redis commands can help our systems perform high-performance operations on the data structures with minimal complexity. Recommendation engine scenarios typically require some quick-to-execute operations of Set such as intersection, union, and set difference. We can easily use Redis's Strings,

Sets and Sorted Sets data structures to implement the recommendation engine or an in-memory database platform. Redis achieves sub-millisecond performance with minimal computing resources. For real-time recommendations, the final product we need is to recommend a set of items for each user or a Sorted Set. As a high-performance, low-latency, in-memory DBMS, Redis can usually do the calculations required for good recommendations.

3. Why use HBase instead of Cassandra?

In the recommendation system, we chose Redis as our online database. But memory is expensive and limited, and we can't store all the data in Redis, so we also need an offline NoSQL database that stores all the data. After a lot of research, we chose the former in the two mainstream recommendation system databases HBase and Cassandra.

HBase is an open source distributed storage system. It can be seen as an open source implementation of Google's Bigtable. Just like Google's Bigtable uses Google File System, HBase is built on top of Hadoop HDFS like Google File System.

Cassandra can be seen as an open source implementation of Amazon Dynamo. The difference with Dynamo is that Cassandra combines the data model of Google Bigtable's Column-Family. It can be simply assumed that Cassandra is a P2P, highly reliable distributed file system with a rich data model.

The following table is a simple comparison table between HBase and Cassandra:

Points	**HBase**	**Cassandra**
CAP Theorem	Consistency & Availability	Availability and Partition Tolerance

Coprocessor	Yes	No
Rebalancing	HBase provides Automatic rebalancing within a cluster.	Cassandra also provides rebalancing but not for overall cluster
Architecture Model	It is based on Master-Slave Architecture Model	Cassandra is based on Active-Active Node Modal
Base of Database	It is based on Google BigTable	Cassandra is based on Amazon DynamoDB
SPoF (Single Point of Failure)	If Master Node is not available entire cluster will not be accessible	All nodes having the same role within cluster so no SPoF
DR (Disaster Recovery)	DR is possible if Two Master Nodes are configured.	Yes, as all nodes having the same role
HDFS Compatibility	Yes, As HBase stores all meta-data in HDFS	No
Consistency	Strong	Not Strong as HBase

HBase vs Cassandra Comparison Table

Facebook & another social networking side would prefer HBase (earlier both were using Cassandra) because of its availability other side banking domain sector looks for security for its

every financial transaction so they would select Cassandra over HBase. Cassandra Key characteristics involve High Availability, Minimal administration and No SPoF other side HBase is good for faster reading and writing the data with linear scalability.

In our recommendation system, both databases can be selected. But in order to take full advantage of the distributed storage capabilities provided by HDFS, we prefer to choose HBase. In addition, compared to Cassandra's key-value pair model, the column family model provided by HBase allows us to have more flexible settings for the number of versions of user information saved, so that we can read data easily for offline model creation. So we finally adopted HBase.

3.3.2 Databases

3.3.2.1 Tables in Redis

1. `User recent interaction table`

Redis data structure: LIST.

Save the user's interaction information for the last ten or so. For efficiency reasons, the method of randomly clearing expired logs is adopted, so the number of logs retained by different users may be slightly different.

Redis data format: latest: userid

E.g., for user 12345, his most recent log is stored in the key "latest:12345".

2. `User recommended item list`

Redis data structure: LIST.

Save a specified number of recommended items recommended to the specified user.

Redis data format: rec:userid.

E.g., ror user 12345, the latest recommended items is saved in "rec: 12345".

3. Recall Items Table

Redis data structure: HASH.

Save a specified number of items hottest over a specific city, brand, category, etc

Save a specified number of items hottest within a day, a week and a month

Save a specified number of items for a user using CF algorithm

Redis data format: "city:cityid", "hottest_one_day_before"

3.3.2.2 Tables in HBase

1. User information

User personal information, including user id, user age, gender, etc., which is used to build the user model:

Description	**Field Name**	**Data Type**
user id	userId	int
age	age	int
sex	sex	String
timestamp	timestamp	String

user level	userLevel	int
city	city	int
province	province	String
city level	cityLevel	String
country	country	String

2. Item information table

Item information, including item id, brand, shop id, category and market time:

Description	**Field Name**	**Data Type**
item id	itemId	int
brand id	brandId	int
timestamp	timestamp	String
user level	userLevel	int
shop id	shopId	int
category	cate	int

3. User log information

User behavior log information including user id, item id, action time, action type(1.browse; 2. order; 3. add to like; 4. comment; 5.add to shopping cart):

Description	**Field Name**	**Data Type**
user id	userId	int

item id	itemId	int
action time	timestamp	String
action type	type	int

4. System Experimental Analysis

4.1 Dataset

The data set is based on the data set of Jdata of JingDong Data Science Competition in 2019. The URL is https://jdata.jd.com/html/detail.html?Id=8.

The organizer of training data provides users in the user set from 2018-2-1 to 2018-4-15, and evaluates the behavior, evaluation and user data of some commodities in the commodity set.

4.1.1 Preprocessing

The whole behavior data set is 1.65G in size, including user unique identification, item unique identification, behavior time and behavior type.

	File Name	Table	Size
1	jdata_action.csv	Action table	1.65GB
2	jdata_comment.csv	Comment table	41MB
3	jdata_shop.csv	Shop table	540KB
4	jdata_user.csv	User table	74MB
5	Jdata_product.csv	Product table	14MB

In fact, we have compared many data sets before, such as Alibaba, Amazon and so on. But the main reason why we use Jingdong data set is that it has user profile, comment profile, and commodity profile, which is quite complete compared with other data sets. In addition, the timestamp range of the data set is two and a half months, and only part of users and items are extracted from the data set, which is of moderate size. This is very helpful for our experiment.

4.1.2 Build positive & negative records

For data sets, if FM algorithm is to be applied, label will be a two-class problem, clicked or not. But for Jdata datasets, only data clicked by users, not data viewed but not clicked, so it is necessary to generate negative sample data artificially.

For program systems, a blog gives the following suggestions:

Positive samples can be defined as "likes" that the user broadcasts on the same day. Negative samples have two options:

(1) Negative samples refer to programs that have been exposed to users, but users have not played them all, that is to say, the programs are not classified as "history" and "like". (2) Negative sample refers to the whole sample pool, but the user has never played it, that is to say, the program is not in the two categories of "history" and "like".

Obviously, for our e-commerce system, only the second artificial generation method can be used. But the randomness is too strong, so I finally decided: a user clicked only once as a negative sample, clicked twice or more as a positive sample.

Later, considering this, some problems will arise. For example, users will click on some items repeatedly. They all belong to one category, but some things will not click on the second time, and then this item will be negative samples, but this does not mean that users do not like this category of things. So, I finally adopted the random sampling method, that is, what the user did not see, or what did not intersect with the user, is what the user did not like.

Establishment: For each user, create item set that the user has not clicked, and the number of clicks that the user has (because for more users with positive samples, it should have more negative

samples). So the user clicks more, and he generates more negative samples. According to the above method, we will randomly generate negative samples of the same order of magnitude.

4.1.3 Training & Testing Set

We divide the data generated by 4.1.2 into training set and test set according to the time. The training machine adds negative samples to the data of the first two months, and the test set adds negative samples to the data of the last 15 days.

Because we can't do online testing and real-time feedback from users, we use the data of the first two months as the total log in HBase and the data of the next 15 days as the new log of users as the flow to Flink in the system simulation.

The model is trained by a python package *xlearn[16]*, which is a high-performance, easy-to-use and extensible machine learning algorithm library. We can use it to solve large-scale machine learning problems, especially large-scale sparse data machine learning problems.

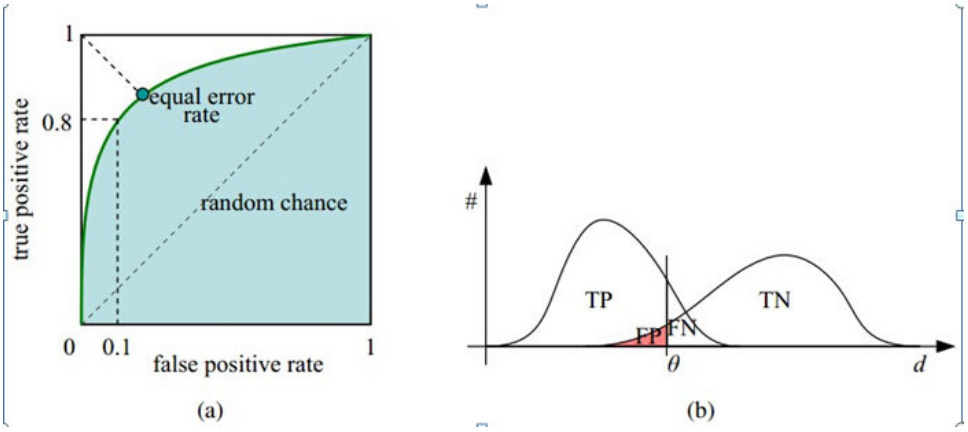
4.2 Model Evaluation

Since there is no real ABTest, the simulation of the effect of our recommendation system will be an offline process. In the next section, we will introduce the method of model evaluation and its effect.

4.2.1 Metrics

CTR (Click-Through-Rate) is a term commonly used in Internet advertising. It refers to the click-through-rate of online advertising, that is, the actual number of clicks (strictly speaking, the number of pages reached) of the advertisement divided by the amount of Show content of the advertisement. CTR is an important index to measure the effect of Internet advertising.

Normally, we usually use CTR as metric. However, because we don't have real-time data online, that is to say, we have no way to know the user's feedback on the products recommended by the system. So we have to change our strategy. As metric, that is, metric for algorithm. For recall algorithm, because the algorithm is rough, there is no comparison to evaluate it. For ranking algorithm, we can measure it by specific metrics. Here we use AUC.



AUC Metric

AUC ordinates: (1) True Positive Rate TPR: TP/(TP+FN) represents the proportion of actual positive instances in the positive classes predicted by the classifier.

AUC abscissa: (2) False Positive Rate FPR: FP/(FP+TN), which represents the proportion of actual negative instances in the positive classes predicted by the classifier.

AUC value is a probability value. When you randomly select a positive sample and a negative sample, the probability that the current classification algorithm ranks the positive sample before the negative sample according to the calculated Score value is AUC value. The larger the AUC value, the more likely the current classification algorithm will rank the positive sample before the negative sample, so that it can better classify. 。

Now that there are so many standards, why use ROC and AUC? Because the ROC curve has a good characteristic: when the distribution of positive and negative samples in the test set is changed, the ROC curve can remain unchanged. In the actual data set, the sample class imbalance often occurs, that is, the proportion difference between positive and negative samples is large, and the positive and negative samples in the test data may change with time.

4.2.2 Model Selection

Before we finally determine the ranking model, we conduct model comparison experiments. The main models used are LR, FM, FFM and LightGBM. LightGBM is not listed in the table because its AUC is always 1 which has some problems.

For FM and FFM models, we adjust their ƛ and *k* parameters. It can be seen that FFM needs less k than FM, and the model of FFM is much larger than FM.

Model	Parameter	Training Time(sec)	Model Size	AUC
LR		9.08	8.38 MB	0.60288
FM	ƛ=0.0002, k=16	18.64	142.48 MB	0.75949
FM	ƛ=0.0002, k=64	47.96	544.79 MB	0.75985
FM	ƛ=0.002, k=16	24.12	142.48 MB	0.74017
FM	ƛ=0.002, k=64	54.60	544.79 MB	0.73996
FFM	ƛ=0.0002, k=4	23.30	310.11 MB	0.75932
FFM	ƛ=0.0002, k=6	34.61	611.84 MB	0.75090
FFM	ƛ=0.0002, k=8	34.28	611.84 MB	0.77035

Model	Parameter	Training Time(sec)	Model Size	AUC
FFM	ƛ=0.002, k=4	23.40	310.11 MB	0.75189
FFM	ƛ=0.002, k=6	38.14	611.84 MB	0.74347
FFM	ƛ=0.02, k=4	24.19	310.11 MB	0.74455

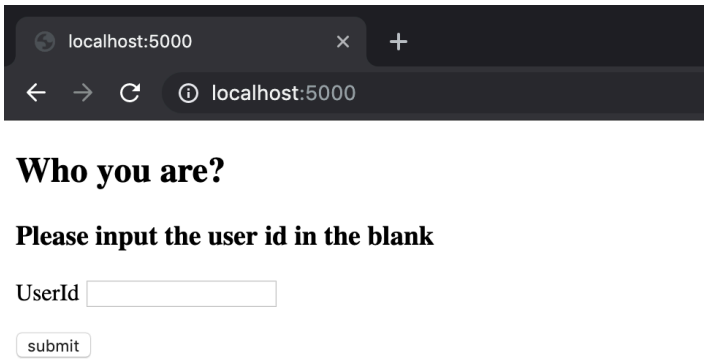
After comprehensive consideration, the AUC of FM and FFM is not much different, but considering the model size and training time and other factors, the final ranking algorithm uses FM.

4.3 Web UI

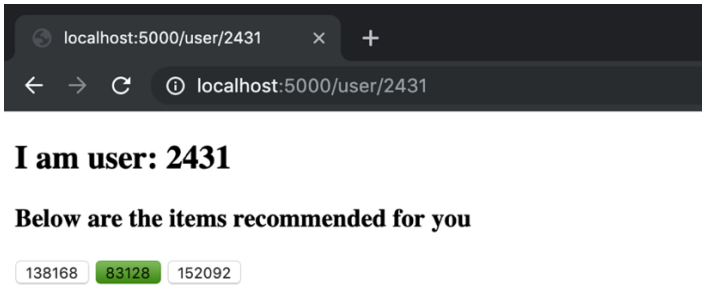
We use *Flask* framework to build our web services. Since our focus is not here, so we only need to schematically complete the basic operations. As shown below, the user ID can be entered on the front page of the interface, which is equivalent to login. After login, if a new user is detected, a cold start mechanism will be activated to return to the current hottest commodity. If it is an old user, it lists the products currently recommended to that user through the log model stored in the database.

When a user clicks on an item, a thread is triggered to run the online model and return to the next recommendation in a short (real-time) time.

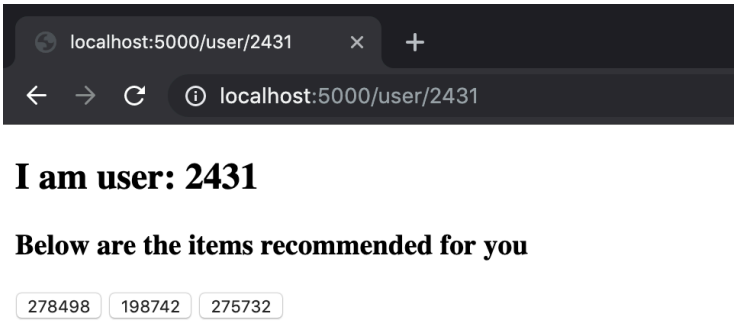
Home Page:



We input 2431 as our user id, and we can see the items recommended. Then we click on 83128:



Then, the recommended items change in real-time:



5. Conclusion

5.1 Conclusion

The emergence of a recommender system provides a powerful solution to the problem of information overload. The recommender system designed in this project has achieved good results in the experiment and should be applied to the actual industrial-level recommender system.

This paper first introduces the background and research significance of the development of the recommender system and then establishes the main research content. Then the concept and input and output of the recommender system are analyzed, the classic recommendation algorithm is introduced, and the advantages and disadvantages of different algorithms are compared, which provides theoretical support for the design and implementation of the system. Then, from the three aspects of system network structure design, system function module design and database design, the outline design of the system is expounded and the overall technical realization scheme is established. Finally, by describing the specific experimental process, the data processing and algorithm parts of our system are elaborated. As for Algorithm, single algorithm has limitations, and it is difficult to recommend for users from multiple perspectives. In the part of algorithm modeling, in recall, collaborative filtering is mainly used to get candidates by establishing similarity matrix of items, and other candidates are obtained by statistics of user characteristics and the most popular items. In order to ensure the smooth operation of the system, the model FM, which trades speed and memory as well as accuracy, is selected after comparison. The comparison of the algorithms and the details of the algorithms are explained in this report.

Finally, we use AUC evaluation metric to test our system. It turns out that we have achieved very good results. At the same time, we design a simple web interface that shows how users interact with a fully transparent recommender system.

5.2 Further Investigations

The real-time e-commerce recommendation system implemented in this paper has completed the preliminary construction, but due to the limited team level and time, there are still many shortcomings in the design and implementation of the entire recommendation system, which need to be further explored and improved in the future work. The following optimization space is as follows:

1. Memory needs to be further optimized, and the current work is to reduce the training features on the premise of reducing accuracy.

2. The recall strategy should be further optimized, especially the collaborative filtering algorithm. The weights of similarity matrix should be selected by more comparisons.

3. Deep learning, as a current research hotspot, should be applied in sorting algorithm.

6. Acknowledgments

Thank our parents for their upbringing and support. Thank our supervisor T.W. CHIM. The guidance he gave us is enlightening. We learned a lot in the experiment process. Thanks for the funds provided by the department, so that we can rent the college's high-quality server to complete our work. Thank the computer science department of the College of Engineering of the University of Hong Kong for its one-year courses and teachers. Each course has laid the foundation for us in different ways.

Thank my teammates, although our team has only two people, but our mutual support and cooperation let us successfully complete such a large project.

Appendix

Github addresses:

1. https://github.com/bjwu/Ecom-Recommend-Algo
2. https://github.com/bjwu/Ecommerce-recommender

References

[1] SCHWARTZ C. Web search engines[J]. Journal of the American Society for Information Science, 1998, 49(11): 973-976
[2] G.D. Linden, J.A. Jacobi, and E.A. Benson, Collaborative Recommendations Using Item-to-Item Similarity Mappings, US Patent 6,266,649, to Amazon.com, Patent and Trademark Office, 2001 (filed 1998)
[3] G. Linden, B. Smith, and J. York, Amazon.com Recommendations: Item-to-Item Collaborative Filtering, IEEE Internet Computing, vol. 7, no. 1, 2003, pp. 76–80
[4] K. Chakrabarti and B. Smith, Method and System for Associating Feedback with Recommendation Rules, US Patent 8,090,621, to Amazon.com, Patent and Trademark Office, 2012.
[5] Sanjeevan Sivapalan, Alireza Sadeghian, Hossein Rahnama and Asad M. Madni, Recommender Systems in E-Commerce, IEEE World Automation Congress(WAC),Aug.2014
[6] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li and Kun Gai, Deep Interest Network for Click-Through Rate Prediction, Machine Learning (stat.ML), Jun.2017
[7] Yu Wang, Jixing Xu, Aohan Wu, Mantian Li, Yang He, Jinghe Hu and Weipeng P. Yan, Telepath: Understanding Users from a Human Vision Perspective in Large-Scale Recommender Systems, arXiv:1709.00300,Sep.2017
[8] J. Liu, T. Zhou, and B. Wang. Research progress of personalized recommendation system. Progress in Natural Science, 19(1):1–15, 2009.
[9] P. Resnick, N. Iacovou, etc. GroupLens: An Open Architecture for Collaborative Filtering of Netnews, Proceedings of ACM Conference on Computer Supported Cooperative Work, CSCW 1994. pp.175-186.
[10] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-Based Collaborative Filtering Recommendation Algorithms, Proceeding of the 10th international conference on World Wide Web, WWW 2001.
[11] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In The adaptive web, pages 325–341. Springer, 2007.
[12] Paul Covington, Jay Adams and Emre Sargin, Deep Neural Networks for YouTube Recommendations, Proceedings of the 10th ACM Conference on Recommender Systems, ACM,2016
[13] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li and Kun Gai, Deep Interest Network for Click-Through Rate Prediction, Machine Learning (stat.ML), Jun.2017
[14] Rendle S, Gantner Z , Freudenthaler C , et al. Fast context-aware recommendations with factorization machines[J]. Proc of Acm Sigir, 2011:635-644.
[15] Yuchin Juan , Yong Zhuang , Wei-Sheng Chin , Chih-Jen Lin, Field-aware Factorization Machines for CTR Prediction, Proceedings of the 10th ACM Conference on Recommender Systems, September 15-19, 2016, Boston, Massachusetts, USA
[16] xlearn documentation. https://xlearn-doc.readthedocs.io