7 Evaluation in Recommender Systems

The proportion of knowledge related to the evaluation of the recommender system is not large in the entire recommender systems knowledge framework, but its importance is as significant as building a recommender system. The evaluation mainly includes the following three points:

(1) The metrics used in the evaluation of the recommender systems directly determine whether the optimization of the recommendation system is objective and reasonable.
(2) The evaluation is a collaborative effort, which requires the machine learning team to communicate and cooperate with other teams.
(3) The selected metrics directly determine whether the recommender system meets the company's business goals and development vision.

These three points are the keys to the success of a recommender system.

This chapter focuses on the evaluation of recommender systems, from offline evaluation to online experiment. It discusses the methods and metrics of recommendation system evaluation from multiple levels, including the following:

(1) Offline evaluation methods and metrics.
(2) Offline simulation evaluation – replay.
(3) Online A/B testing and online metrics.
(4) Fast online evaluation method – interleaving.

These evaluation methods are not independent. At the end of this chapter, we will discuss how to combine different levels of evaluation methods to form a scientific and efficient multilayer recommender system evaluation architecture.

7.1 Offline Evaluation Methods and Basic Metrics

In the evaluation process of the recommender system, offline evaluation is often treated as the most common and basic evaluation method. As the name suggests, offline evaluation refers to the evaluation performed in an offline environment before deploying the model online. Since there is no need to deploy to the production environment, offline evaluation does not have the engineering risk of online deployment, and there is no need to waste valuable online traffic resources. Additionally, it has many other advantages, such as short test time, multiple parallel tests at the same time, the ability to use abundant offline computing resources, and so on.

Therefore, before the model is launched online, conducting a large number of offline evaluations is the most efficient way to verify the model performance. In order to fully grasp the technical points of offline evaluation, it is necessary to master two aspects of knowledge – the methods and the metrics used in offline evaluation.

7.1.1 Methods of Offline Evaluations

The basic principle of offline evaluation is to divide the dataset into a training set and a testing set in an offline environment. The training set is used to train the model, and the testing set is for model evaluation. According to different dataset partition methods, offline evaluation can be divided into the following three types:

7.1.1.1 Holdout Test

The holdout test is a basic offline evaluation method, which randomly divides the original sample set into two parts – the training set and the testing set. For example, for a recommendation model, the samples can be randomly divided into two parts according to the ratio of 70%:30%, where 70% of the samples are used for model training and 30% of the samples are used for model evaluation.

The disadvantage of the holdout test is obvious. The evaluation metric calculated on the testing set is directly related to the division of the training set and the testing set. If only a small amount of holdout test is performed, the conclusions obtained will be relatively random. In order to eliminate this randomness, the idea of cross-validation is proposed.

7.1.1.2 Cross-Validation

- K-fold cross-validation. In this method, the samples are firstly divided into k partitions. Then, the training and evaluation process will traverse these k subsets in turn. In each turn, we use the current subset as a testing set and use all other subsets as a training set to perform model training. Finally, the average of all the evaluation metrics for k runs is used as the final evaluation result. In practical experiments, k is usually equal to 10.
- Leave-one-out validation. One sample is left as a testing set each time, and all other samples are used as the training set. Assuming the total number of samples is n, all n samples are traversed in turn. Then, n times of evaluation are performed, and the evaluation metrics are averaged to obtain the final performance result. In the case of a large number of samples, the time overhead of the leave-one-out validation method is extremely high. In fact, leave-one-out validation is a special case of leave-p validation. Leave-p validation means leaving p samples as the testing set each time, and there are C_n^p possibilities to select p elements from n elements, so its time complexity is much higher than that of leaving one verification. As a result, it is rarely used in practice.

7.1.1.3 Bootstrap

Both the holdout test and the cross-validation are based on the method of dividing whole datasets into training and testing sets. However, when the sample size is relatively small, sample set division will further reduce the training sample amount, which may affect the training effect of the model. Is there an evaluation method that can maintain the sample size of the training set? The bootstrap approach can solve this problem to a certain extent.

Bootstrap is a test method based on the resampling technique. For a sample set with size of n, random sampling with replacement is performed n times to obtain a training set with size of n. In the n-time sampling process, some samples will be re-sampled, and some samples will not be drawn. The bootstrap method uses these undrawn samples as a testing set for model evaluation.

7.1.2 Offline Evaluation Metrics

After knowing the correct offline evaluation method, to measure the performance of a recommendation model, it is necessary to evaluate the recommender system from multiple perspectives through different metrics and then draw comprehensive conclusions. The following are metrics that are commonly used in offline evaluation.

7.1.2.1 Accuracy

Classification accuracy refers to the ratio of correctly classified samples against the total number of samples, that is,

$$\text{Accuracy} = \frac{n_{\text{correct}}}{n_{\text{total}}} \tag{7.1}$$

where n_{correct} is the number of correctly classified samples and n_{total} is the total sample count.

Accuracy is a relatively intuitive evaluation metric in classification tasks. Although it has strong interpretability, it also has drawbacks. When the proportion of samples in different categories is very imbalanced, the category with a large proportion often becomes a factor that affects the accuracy rate. For example, if negative samples account for 99%, then the classifier can predict all samples as negative samples to obtain 99% accuracy.

For a click-through rate prediction classification problem, the recommendation model can be evaluated with the accuracy rate under the premise of selecting a threshold to determine positive and negative samples. In the actual recommendation scenario, the more common use case is to generate a recommendation list. So the combination of precision and recall are more commonly used to measure the performance of recommendations.

7.1.2.2 Precision and Recall

Precision is the ratio of the number of correctly classified positive samples against the number of predicted positive samples, while recall is the ratio of the number of correctly classified positive samples to the number of true positive samples.

In the ranking model, there is usually no definite threshold to directly judge the prediction result as a positive sample or a negative sample. The precision rate (Precision@N) and recall rate (Recall@N) of the Top N-ranked results are usually used to evaluate the ranking model's performance. In this case, the Top N items are considered the positive samples predicted by the model in the precision rate and recall rate calculation.

Precision rate and recall rate are contradictory indicators. In order to improve the precision rate, the model needs to predict the sample as a positive sample when it has high confidence, but it often misses many true positive samples when the model is not so confident, which results in a lower recall rate.

In order to comprehensively reflect the results of precision and recall, the F1-score is often adopted. F1-score is the harmonic mean of precision and recall, which is defined as follows:

$$\mathrm{F1} = \frac{2 \cdot \mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{7.2}$$

7.1.2.3 Root Mean Square Error

Root mean square error (RMSE) is often used to measure the quality of the regression model. When using the click-through rate prediction model to build a recommender system, the recommender system actually predicts the probability of a positive sample. It can be evaluated by RMSE, which is defined as follows,

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2}{n}} \tag{7.3}$$

where y_i is the ground truth label of i-th sample, $\hat{y}_i$ is the predicted value of the i-th sample, and n is the number of samples.

In general, RMSE can well reflect the degree of deviation between the predicted value of the regression model and the true value. However, in practical applications, if there are individual outliers with a very large degree of deviation, the RMSE can become quite large even if the number of outliers is small. To solve this problem, mean absolute percent error (MAPE) is often adopted to improve the robustness against outliers. The definition of MAPE is as follows,

$$\mathrm{MAPE} = \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times \frac{100}{n} \tag{7.4}$$

Compared with RMSE, MAPE is equivalent to normalizing the error of each sample, which reduces the impact of absolute error brought by individual outliers.

7.1.2.4 Logarithmic Loss Function

Logarithmic loss function (LogLoss) is another metric that is often used in offline evaluation. In a binary classification problem, LogLoss can be defined as follows,

$$\mathrm{LogLoss} = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log P_i + \left(1 - y_i\right) \log\left(1 - P_i\right) \right) \tag{7.5}$$

Among them, y_i is the ground truth label of the sample x_i, P_i is the probability of predicting that the input sample x_i is a positive sample, and N is the total number of samples.

Readers could find that LogLoss is the loss function of logistic regression. A large number of deep learning models use logistic regression (that is, Sigmoid) or Softmax as the output layer. Therefore, using LogLoss as an evaluation metric can very intuitively reflect the change of the model's loss function. From the perspective of the model, LogLoss is a very suitable evaluation metric for the model's convergence.

7.2 Offline Metrics for Ranking Models

Section 7.1 introduces the main offline evaluation methods and common evaluation metrics of the recommender system, but they are more commonly used as the evaluation of a prediction model such as click-through rate (CTR) prediction, rather than a ranking model. Usually, we hope the outputs of the prediction should have a physical meaning. In the CTR prediction case, the model output should be close to the empirical click-through rate. However, the final output of the recommender system is usually a ranked list. Taking the matrix decomposition method as an example, the similarity between users and items is only a criterion used for sorting, and does not have a physical meaning like CTR. Therefore, it is more appropriate to evaluate recommendation models using metrics that can directly measure the ranking quality. In this section, we will introduce several offline metrics for directly measuring the ranking model performance. These metrics are Precision-Recall (P–R) curve, Receiver Operating Characteristic (ROC) curve, and mean average precision (mAP).

7.2.1 Precision–Recall Curve

Section 7.1 introduces two important metrics for evaluating sorted sequences, Precision@N and Recall@N. In order to comprehensively evaluate the quality of a ranking model, we should not only check the Precision@N and Recall@N of the model under different Top N, but also to draw the Precision-Recall curve (P–R curve) of the model. In this section, we will briefly introduce the method of generating P–R curve.

The horizontal axis of the P–R curve is the recall rate, and the vertical axis is the precision rate. For a ranking model, a point on its P–R curve represents the precision and recall for a given threshold. The threshold is used to determine if prediction output is positive or negative. In other words, if the model prediction output is greater than the threshold, then it is predicted as positive sample, otherwise it is considered as negative sample.

The entire P–R curve is generated by changing the threshold from high to low. As shown in Figure 7.1, the solid line represents the P–R curve of model A, and the dotted line represents the P–R curve of model B. The areas near horizontal axis origin represents the precision and recall of the model when the threshold is maximum.

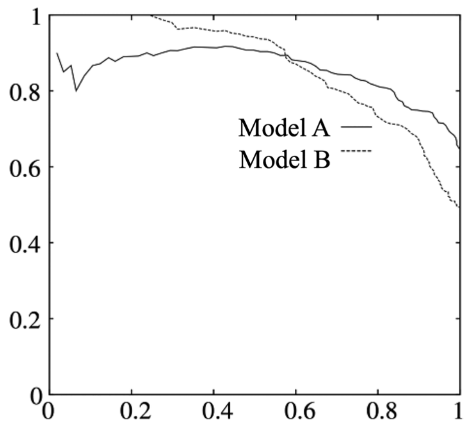

Figure 7.1 Examples of P–R curves.

It can be seen from Figure 7.1 that when the recall rate is close to 0, the precision rate of model A is 0.9, and the precision rate of model B is 1. This implies that the samples with the top scores in model B are all true positive samples, while model A has some prediction errors even when the predicted score is quite high. As the recall rate increases, the precision rate decreases overall. In particular, when the recall rate is 1, the precision rate of model A exceeds that of model B. This demonstrates that the performance of the model cannot be fully measured by using the precision and recall of a single point. The overall model performance should be evaluated using a comprehensive metric such as the P–R curve.

After the P–R curve is generated, we can use the area under the curve (AUC) to quantify the P–R curve. As the name implies, AUC refers to the area under the P–R curve, so calculating the AUC value needs to be integrated along the horizontal axis of the P–R curve. The larger the AUC, the better the ranking model's performance.

7.2.2 Receiver Operating Characteristic Curve

The ROC curve was first introduced in the military field. Then, it was widely used in the medical field, and that's also where the ROC concept in the machine learning domain originated.

The horizontal axis of a ROC curve is the False Positive Rate (FPR) and the vertical axis is the True Positive Rate (TPR). FPR and TPR are defined as follows,

$$\text{FPR} = \frac{\text{FP}}{N}, \text{TPR} = \frac{\text{TP}}{P} \tag{7.6}$$

where P is the number of positive samples, N is the number of negative samples, TP denotes the number of positive samples that the model correctly predicts the positive class, and FP refers to the number of negative samples that are wrongly predicted as positive by the model.

Table 7.1 An example of ranking model prediction outputs

Sample #	Ground Truth Label	Model Predictions	Sample #	Ground Truth Label	Model Predictions
1	P	0.9	11	P	0.4
2	P	0.8	12	N	0.39
3	N	0.7	13	P	0.38
4	P	0.6	14	N	0.37
5	P	0.55	15	N	0.36
6	P	0.54	16	N	0.35
7	N	0.53	17	P	0.34
8	N	0.52	18	N	0.33
9	P	0.51	19	P	0.30
10	N	0.505	20	N	0.1

The definition of the ROC curve seems complicated, but the process of generating the ROC curve is not difficult. Next, we will show how to draw a ROC curve with an example and let readers understand how an ROC curve is used to evaluate the ranking model performance.

Like the P–R curve, the ROC curve is generated by continuously changing the model's positive sample prediction threshold. Here is an example to explain the process.

Assuming that there are 20 samples in the test set, the output of the model prediction is shown in Table 7.1. The first column in the table is the sample index, the second column is the ground truth label of the sample, and the third column is the probability that the model predicts as positive. Samples are sorted from highest to lowest predicted probability. Before deciding the final positive and negative examples, a threshold needs to be specified: samples with a predicted probability greater than the threshold will be considered as positive examples, and samples smaller than the threshold will be considered as negative examples. If the threshold is 0.9, then only the first sample will be predicted as a positive example, and all others will be negative examples. The threshold here is also known as the cut-off point.

Then the threshold is changed to different values, starting from the highest score (actually starting from positive infinity, corresponding to the zero point of the ROC curve), gradually to the lowest score. At each threshold, it generates to a pair of FPR and TPR values. The value pairs are then plotted correspondingly on the ROC graph. Finally, the ROC curve is obtained by connecting all the data points sequentially on the graph.

For this example, when the threshold value is positive infinity, the model predicts all samples as negative examples. Both FPR and TPR are also 0. So the first point of the curve is (0,0). When the threshold is changed to 0.9, the model predicts that sample #1 is a positive sample, and the sample is a real positive example. Therefore, TP=1. In 20 samples, the number of all positive cases is P=10, so TPR=TP/P=1/10. In this example, there is no positive sample that is wrongly predicted, that is, FP=0, and

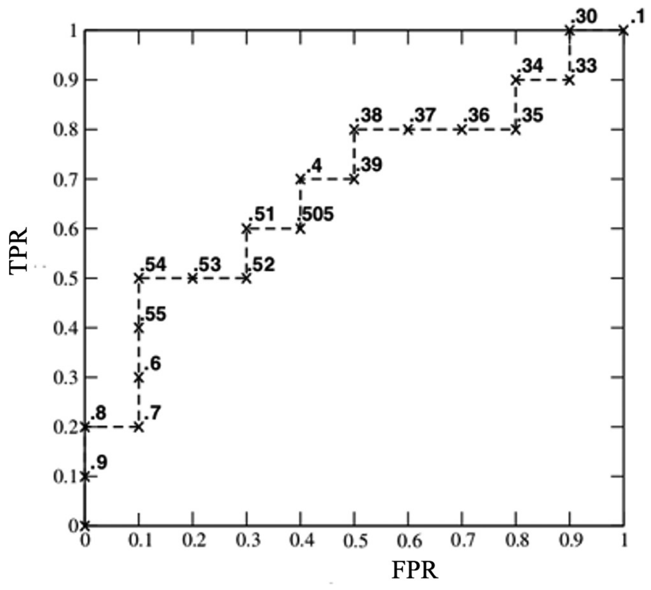

Figure 7.2 ROC curve.

the total number of negative samples is N=10, so FPR=FP/N=0/10=0. It corresponds to the point (0,0.1) on the ROC diagram. Continue changing the threshold value in turn until all the key points are drawn, and then connect the points to get the final ROC curve, as shown in Figure 7.2.

In fact, there is a more intuitive way to draw the ROC curve. First, count the number of positive and negative samples according to the sample label, assuming that the number of positive samples is P and the number of negative samples is N. Next, set the interval of the horizontal axis to 1/N, and the interval of the vertical axis to 1/P Then, sort the samples according to the predicted probability output by the model (from high to low). Traverse the samples in turn and draw the ROC curve starting from 0 point. Draw a unit interval curve along the vertical axis every time a positive sample is encountered, and a unit interval curve along the horizontal axis for each negative sample. Finally, connect the last point with (1,1), and the entire ROC curve is drawn.

Similar to the P–R curve, the AUC can be calculated after ROC curve is generated. The AUC can be used to evaluate the ranking model's performance in the recommender system.

7.2.3 Mean Average Precision

Mean average precision (mAP) is another commonly used evaluation metric in recommender systems and information retrieval. This metric is actually an average of

Average Precision (AP). Before calculating mAP, readers need to understand what average precision is.

Assume that the ranking results of a user test set by the recommender system are shown in Table 7.2, where 1 represents the positive sample, and 0 represents the negative sample.

In the previous section, we introduced how to calculate precision@N. Then what is the precision@N at each position of this ranking list? The results are shown in Table 7.3.

The calculation of AP only takes the precision for different topN for average calculation, that is, $AP = (1/1 + 2/4 + 3/5 + 4/6)/4 = 0.6917$. How about mAP?

If the recommender system sorts the samples of each user in the test set, then we can get an AP value for each user. The average AP value of all users is then the mAP value.

It is worth noting that the calculation method of mAP is completely different from the calculation methods of the P–R curve and the ROC curve, because mAP needs to sort the samples for each user, while both P–R curve and ROC curve can be calculated with the sorted full test set. This difference needs special attention in the actual calculations.

7.2.4 Selecting Reasonable Evaluation Metrics

In addition to three commonly used metrics like P–R curve, ROC curve, and mAP, there are many other metrics used in the recommender system evaluation, such as Normalized Discounted Cumulative Gain (NDCG), coverage, and diversity, and so on. In the actual offline experiment, although it is necessary to evaluate the model from different angles, there is no need to pursue perfection to find the "best" metric. Choosing too many metrics to evaluate the model could sometimes result in a waste of time. The purpose of offline evaluation is to quickly detect the issues, eliminate unreliable candidates, and find promising candidates for online evaluation. Therefore, selecting two to four representative offline metrics based on the business scenarios and conducting efficient offline experiments is the correct path for offline evaluation.

Table 7.2 Example of ranking results

Ranking List	$N=1$	$N=2$	$N=3$	$N=4$	$N=5$	$N=6$
Ground Truth Label	1	0	0	1	1	1

Table 7.3 Examples of precision@N calculation

Ranking List	$N=1$	$N=2$	$N=3$	$N=4$	$N=5$	$N=6$
Ground Truth Label	1	0	0	1	1	1
Precision@N	1/1	1/2	1/3	2/4	3/5	4/6

7.3 Replay: An Offline Evaluation Method Aligned with the Online Environment

The first two sections introduce the offline evaluation methods and commonly used evaluation metrics in the recommender system. Traditional offline evaluation methods have been widely used in various model experiments in academia. However, during the model development in industry, can these methods (such as holdout test and cross-validation) really objectively measure the model's impact on the company's business goal?

7.3.1 Logical Loop for Model Evaluation

To answer this question, it is necessary to revisit the core of model evaluation – how to evaluate a model and determine whether it is a "good" model? Figure 7.3 shows the logical relationship of each component in the model evaluation.

The key point of offline evaluation is to make the results of offline evaluation as close as possible to online ones. To achieve this goal, the offline evaluation process should simulate the online environment as much as possible. The online environment includes not only the online data environment, but also production settings such as model update frequency.

7.3.2 Dynamic Offline Evaluation Method

The disadvantage of the traditional offline evaluation method is that the evaluation process is static. In other words, the model is not updated with the evaluation, which does not reflect the actual condition in production. Assuming that a recommendation model is evaluated with one month's test data, if the evaluation process is static, it means that when the model predicts the data close to the end of the month, the model has stopped updating for nearly 30 days. This is not practical in most industrial applications. It will lead to the drifting in the model evaluation. In order to solve this

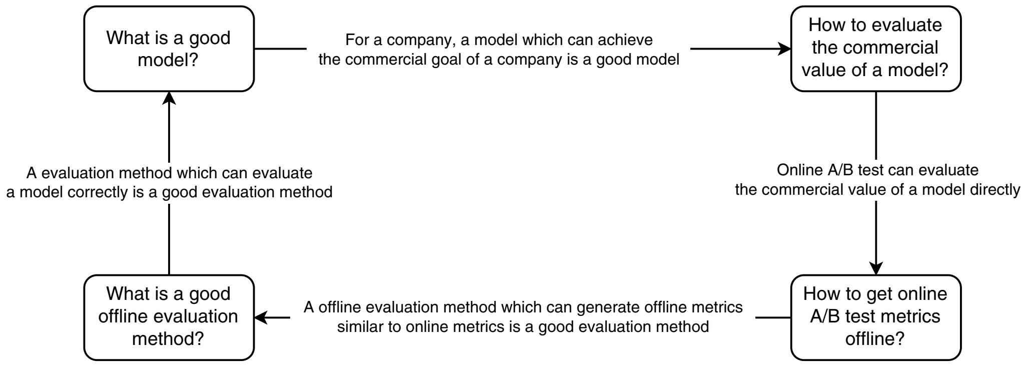
What is a good model?
For a company, a model which can achieve the commercial goal of a company is a good model
How to evaluate the commercial value of a model?
Online A/B test can evaluate the commercial value of a model directly
How to get online A/B test metrics offline?
A offline evaluation method which can generate offline metrics similar to online metrics is a good evaluation method
What is a good offline evaluation method?
A evaluation method which can evaluate a model correctly is a good evaluation method

Figure 7.3 Logical relationship of each component in the model evaluation.

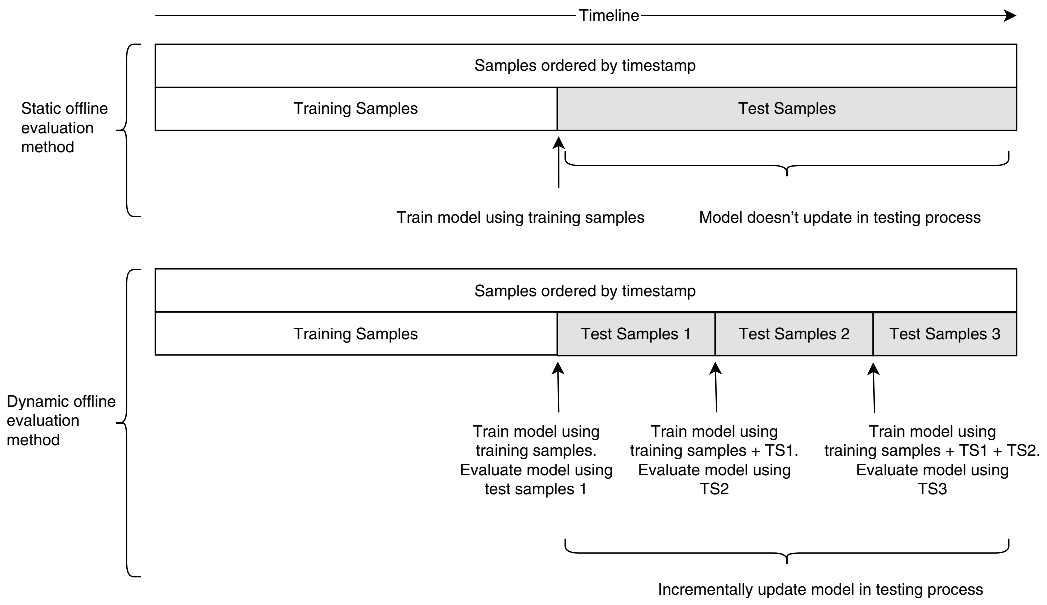
Timeline
Static offline evaluation method
Samples ordered by timestamp
Training Samples
Test Samples
Train model using training samples
Model doesn't update in testing process
Dynamic offline evaluation method
Samples ordered by timestamp
Training Samples
Test Samples 1
Test Samples 2
Test Samples 3
Train model using training samples. Evaluate model using test samples 1
Train model using training samples + TS1. Evaluate model using TS2
Train model using training samples + TS1 + TS2. Evaluate model using TS3
Incrementally update model in testing process

Figure 7.4 Comparison of traditional offline evaluation method and dynamic offline evaluation method.

problem, the entire evaluation process needs to be dynamic to make it closer to an online environment.

The dynamic offline evaluation method first sorts the test samples into chronological order, and then uses the model to predict the test samples in the subsequent time interval. At the time point when the model is updated, the model needs to incrementally learn from the samples before the time point of the model update, and perform its subsequent evaluation after the update. The comparison between the traditional offline evaluation method and the dynamic offline evaluation method is shown in Figure 7.4.

It is easy to see that the dynamic evaluation process is closer to the real online environment, and the evaluation results are closer to the objective situation. If the frequency of model updates continues to increase, the entire dynamic evaluation process becomes an online simulation process of sample playback one by one. This is the classic simulation offline evaluation method – replay.

In fact, replay is the only offline evaluation method for reinforcement learning models [1]. Taking the DRN model introduced in Section 3.10 as an example, since the model needs to continuously receive online feedback and update it online, the replay method must be used offline to simulate the model's online feedback loop and update process.

7.3.3 Replay Evaluation Method Adopted in Netflix

The replay method performs offline testing by replaying online data streams. The principle of the evaluation method is not difficult to understand, but it will encounter some difficulties in actual engineering. The most critical point is that the samples used in a replay cannot contain any "future information" in the simulation data stream. This is to avoid the phenomenon called "data leakage."

For example, the replay method uses the sample data from August 1st to August 31st for replay, and the prediction model has historical CTR as one feature. The calculation of this feature can only be generated through historical data. More specifically, the sample on August 20th can only use the data from August 1st to August 19th to generate historical CTR features and cannot use the data after August 20th. In the evaluation process, if all sample data from August 1st to August 31st were used to generate features for engineering simplification, and then the replay method was used for evaluation, the evaluation result would not be useful since the model uses future knowledge to predict backward samples.

In engineering implementation, Netflix built a complete set of data pipeline architecture (as shown in Figure 7.5) to support the replay evaluation method and gave it a very beautiful name – Time Machine.

It can be seen from the figure that the Time Machine runs once a day. The main function of its primary task, Snapshot Jobs, is to integrate various logs, features, and data of the day to form sample data for model training and evaluation of the day. The date is used in the directory name, and the sample data is stored in the distributed file

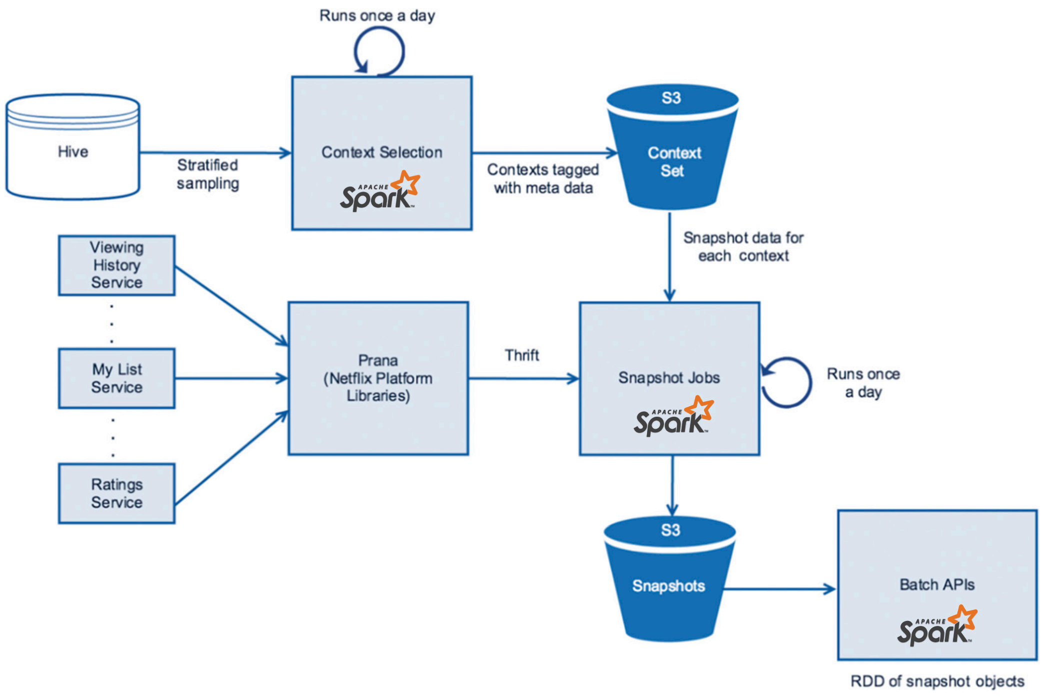

Figure 7.5 Netflix's offline evaluation data pipeline – Time Machine.
Apache Spark, Spark, the Apache Spark Logo and Apache are either registered trademarks or trademarks of the Apache Software Foundation.

system S3. A unified API is provided to external consumers, and these data snapshots can be fetched based on the requested time frame.

From the input of Snapshot Jobs, the information integrated by the Time Machine includes two parts:

(1) Context: it includes relatively static information stored in Hive, such as user profile, device information, item information, and so on.
(2) System log stream: it refers to the logs generated by the system in real time, including the user's viewing history, recommendation impression and user ratings. These logs are generated from different services, and are ingested through Netflix's unified data API – Prana.

Snapshot Jobs retrieve the context information from the context S3 bucket and logs through Prana, then saves the one day's data snapshot to S3 after data processing and feature generation.

After generating the daily data snapshot, it is no longer difficult to use the replay method for offline evaluation. This is because there is no need to perform heavy feature generation during the replay process, and the data snapshot information of the day can be directly used.

On the other hand, it loses flexibility to some extent. Since different models use different features, it is impossible for Snapshot Jobs to generate the required features for all models at once. If some special features are required for a particular model, the Time Machine needs to generate another snapshot for this model based on the common data snapshot.

Based on the framework of the Time Machine, using samples of a certain period of time to perform a replay evaluation is equivalent to having a time travel to this period of time. We hope readers can find their ideal model in this "wonderful" time travel.

7.4 A/B Test and Online Evaluation Metrics

No matter how closely the offline evaluation simulates the online environment, it is impossible to completely reproduce all the online conditions. For almost all internet companies, online A/B testing is the main testing method to validate the effectiveness of new components, new features, and new products.

7.4.1 What Is A/B Test?

A/B test, also known as split test or bucket test, is a random experiment. It usually divided the test group into control (A) and treatment (B). By varying a single variable, it compares the performance of the control and treatment groups correspondingly, and then draws the experiment conclusions based on the collected performance metrics. Specific to the models used in the internet applications, users can be randomly divided into control and treatment groups. Then, the new model is applied to the users in the treatment group, and the old model is applied to the users in the control group.

With some data collection and analysis, the experimenter can get comparisons on the selected online metrics.

Compared with offline evaluation, there are three main reasons why online A/B testing cannot be skipped:

- Offline evaluation cannot completely eliminate the impact of data bias.
- Offline evaluation cannot fully reproduce the online condition. Generally speaking, offline evaluation often does not consider the data latency, data loss, label missing, and so on. Therefore, the offline evaluation results often have some deviation from the reality.
- Some business metrics of the online system cannot be calculated in the offline evaluation. Offline evaluation generally evaluates the model itself, and cannot directly obtain other metrics related to the business target. Taking the new recommendation model as an example, offline evaluation often focuses on improving the ROC curve and PR curve, while online evaluation can fully understand the changes in user click rate, retention time, PV visits, and so on. These metrics can be only obtained through online A/B testing.

7.4.2 Bucketing Mechanism in A/B Testing

In the process of A/B test bucketing, it is necessary to consider the independence of the samples and the unbiasedness of the sampling method. The same user can only be allocated into the same bucket during the entire test, and the user bucketing should be purely random to ensure that the samples in the bucket are not biased.

In the actual industrial online experiment scenario, the website or application often needs to conduct multiple sets of different A/B tests at the same time, such as performing A/B tests on different App UIs at the front end, different middleware efficiencies at the business layer and different algorithms in the recommender system. To avoid the interference of A/B tests at different levels, an effective testing principle must be formulated. Otherwise, the evaluation results can be polluted and misleading caused by improper experiment traffic division. Google's paper [2] introduced in detail the mechanism of experimental traffic layering and partitioning to ensure the high availability of valuable online test traffic.

The mechanism of A/B testing layering and partitioning can be briefly described by two principles:

(1) Orthogonal traffic between layers
(2) Mutually exclusive within the same layer

The orthogonal traffic means that the traffic of each experiment group in the experiment will be randomly partitioned again in the different experiment layers, and evenly distributed in each experiment group in the next layer.

Taking Figure 7.6 as an example, the traffic flow is randomly and equally divided into two parts, X_1 (blue) and X_2 (white) in the Layer X experiment. In the Layer Y experiment, the traffic of X_1 and X_2 should be randomly and evenly distributed to

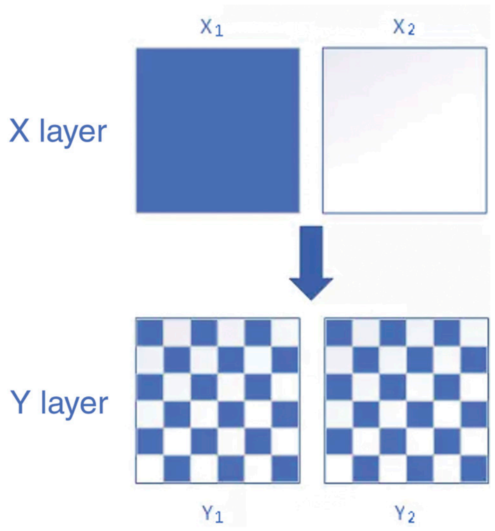

Figure 7.6 Example of orthogonal traffic between layers.

the two buckets Y_1 and Y_2. If the traffic re-distribution of Y_1 and Y_2 is imbalanced in Layer X, then the samples of Layer Y will be biased. As a result, the experimental results of Layer Y will be affected by Layer X. Therefore, the traffic passing through Layer X should be re-randomized and evenly distributed in Y_1 and Y_2.

The meaning of mutually exclusive traffic in the same layer is as follows:

(1) If multiple sets of A/B tests are performed in the same layer, the traffic between different tests should not overlap.
(2) In a group of A/B tests, the traffic of the treatment group and the control group do not overlap.

In user-based A/B testing, "mutual exclusion" means any particular user should only exist in a single treatment group per experiment. Especially for the recommender system, the consistency of user experience is very important, since it usually takes the user some time to adapt to the new recommendation experience. Therefore, it is necessary to ensure that the same user is always assigned to the same group in A/B testing.

The orthogonal and mutually exclusive principles of A/B testing together ensure the objectivity of A/B testing evaluation. Then, how should we choose the evaluation metrics for online A/B testing?

7.4.3 Metrics for Online A/B Testing

Generally speaking, the A/B testing is the last test before the model goes online. The model that passes the A/B test will directly serve the online users to meet the company's business goals. Therefore, the metrics of A/B testing should be consistent with the business key performance indicator (KPI).

Table 7.4 Main evaluation metrics for online A/B testing in various recommender systems

Recommender System Category	Online A/B Metrics
E-commerce	Click-through rate, conversion rate and unit customer spending
News	Retention rate (number of users who are still active after x days/number of total users before x days), average session duration, average number of clicks
Video	Play completion rate (play time/video time), average play time, total play time

Table 7.4 lists the main evaluation metrics for online A/B testing of e-commerce, news and video recommendation models.

Readers should have noticed that the metrics of online A/B testing are quite different from those of offline evaluation (such as AUC, F1-score, and so on). Offline evaluation does not have the conditions to directly calculate the business KPIs, so the next best thing is to choose model-related metrics just for technical evaluation purposes. However, at the company level, there is more interest in the business KPIs that can drive business growth. Therefore, when an online testing environment is available, it is necessary to use A/B testing to verify the effect of the model on improving business KPI. In this case, the role of online A/B testing can never be replaced by offline evaluation.

7.5 Fast Online Evaluation Method: Interleaving

For web applications backed by many recommendation models, in order to continuously iterate and optimize the recommender system, a large number of A/B tests are required to verify the effect of the new algorithm. However, online A/B testing will inevitably take up lots of valuable online traffic resources, and can potentially cause negative impacts to user experience. This generates conflicts between increasing demand for A/B tests and limited resources for online A/B tests.

In response to these problems, a fast online evaluation method – Interleaving – was formally proposed by Microsoft [3] in 2013, and has been successfully applied in production by some companies such as Netflix [4]. Specifically, the Interleaving method is used as the pre-selection stage of the online A/B test, as shown in Figure 7.7, to quickly narrow down the candidate algorithms, and select a small number of "promising" recommendation algorithms from a large number of initial ideas. Then, the traditional A/B testing is performed on the narrowed set of models to measure their long-term impact on user behavior.

In Figure 7.7, light bulbs represent candidate algorithms, with the optimal winning algorithm shown by a red bulb. The Interleaving method can quickly narrow down the initial set of candidate algorithms, determining the best one faster than traditional A/B testing. We'll use Netflix's application scenario as an example to illustrate the principles and characteristics of the Interleaving method.

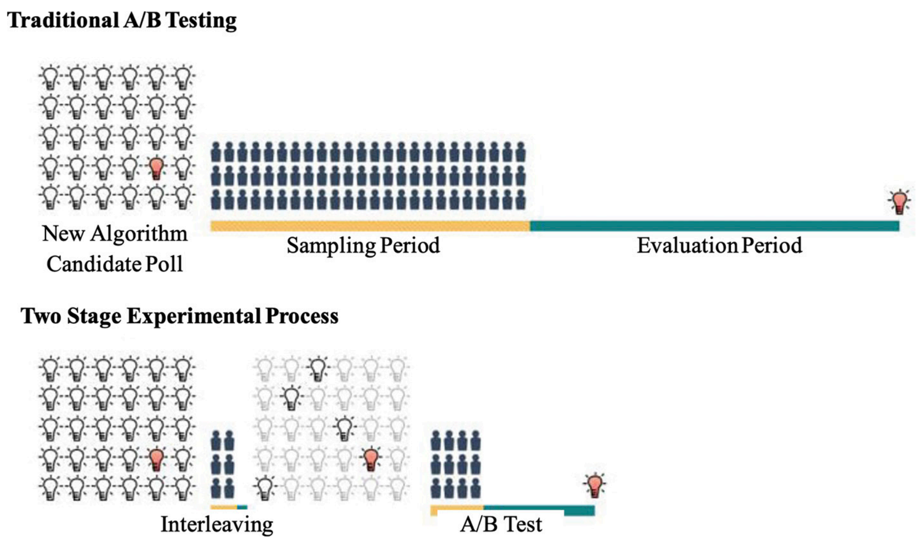

Figure 7.7 Using Interleaving for rapid online testing.

7.5.1 Statistical Issues with Traditional A/B Testing

In addition to efficiency limitations, traditional A/B testing also encounters certain issues with statistical significance. Let's illustrate this with a classic A/B testing example.

Imagine designing an A/B test to assess whether there's a taste preference for Coca-Cola over Pepsi among users. In a traditional setup, participants would be randomly divided into two groups for a blind taste test (where brand labels are hidden). Group A would be given only Coca-Cola, while Group B would only get Pepsi. Consumption over a set period would then indicate a preference for one brand over the other.

While generally effective, this test has potential flaws:

In the test population, consumption habits vary widely, from those who rarely drink soda to heavy daily consumers. Heavy soda drinkers make up a small portion of the sample but may contribute disproportionately to overall consumption. This imbalance could skew results if either group has slightly more heavy consumers, leading to a distorted conclusion.

This issue also arises in online applications like Netflix. A small number of highly active users account for a significant portion of total watch time. So, if more of these active users end up in Group A than in Group B (or vice versa), it can impact the A/B test outcome and obscure the true model performance.

How to address this issue? One solution is to avoid dividing the test population into separate groups. Instead, allow all participants to choose freely between Coca-Cola and Pepsi (while still ensuring the brands remain unlabeled but distinct). At the end of the test, we can calculate each participant's consumption ratio between Coca-Cola and Pepsi, then average these ratios to get an overall preference.

Advantages of this approach include:

(1) It eliminates imbalances in user characteristics between groups.
(2) By assigning equal weight to each participant, it minimizes the impact of heavy consumers on the results.

This approach, where all test options are presented simultaneously to participants and preferences are used to derive evaluation results, is known as the Interleaving method.

7.5.2 Implementing the Interleaving Method

Figure 7.8 illustrates the differences between traditional A/B testing and the interleaving method.

In a traditional A/B test, Netflix will select two groups of subscribers – one group applied with algorithm A, and the other group with algorithm B.

In contrast, in the Interleaving method, there is only one set of subscribers who receive alternate rankings generated by mixing algorithm A and algorithm B.

This allows users to see the recommendation results of both algorithms A and B at the same time in one line (users cannot distinguish whether an item is recommended by algorithm A or B), and then measure the performance of two models using online metrics such as watching time.

While using the Interleaving method for testing, the position bias needs to be considered. The Interleaving method should prevent a particular algorithm from always being ranked first. Therefore, it is necessary to let algorithm A and algorithm B take the lead alternatively with equal probability. This is similar to the process in which the two captains decide who should choose first by tossing a coin and then alternately

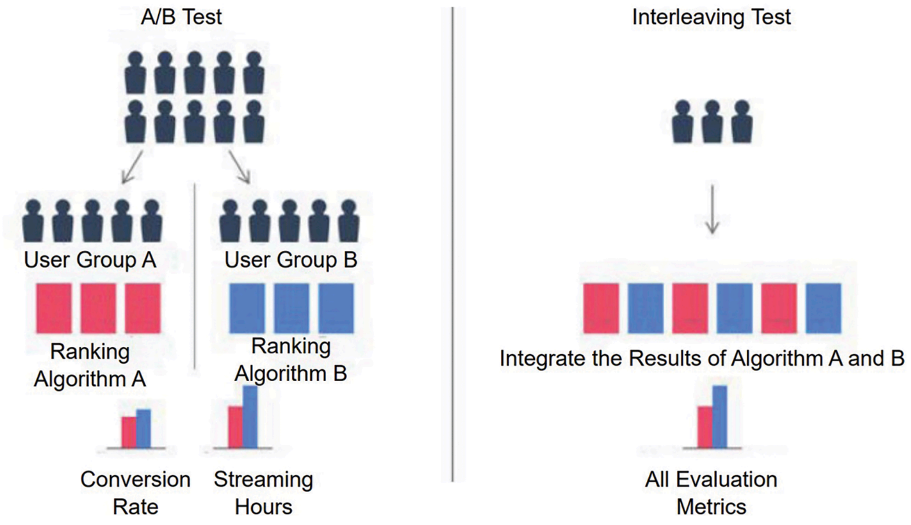

Figure 7.8 Comparison of traditional A/B testing and interleaving method [4].

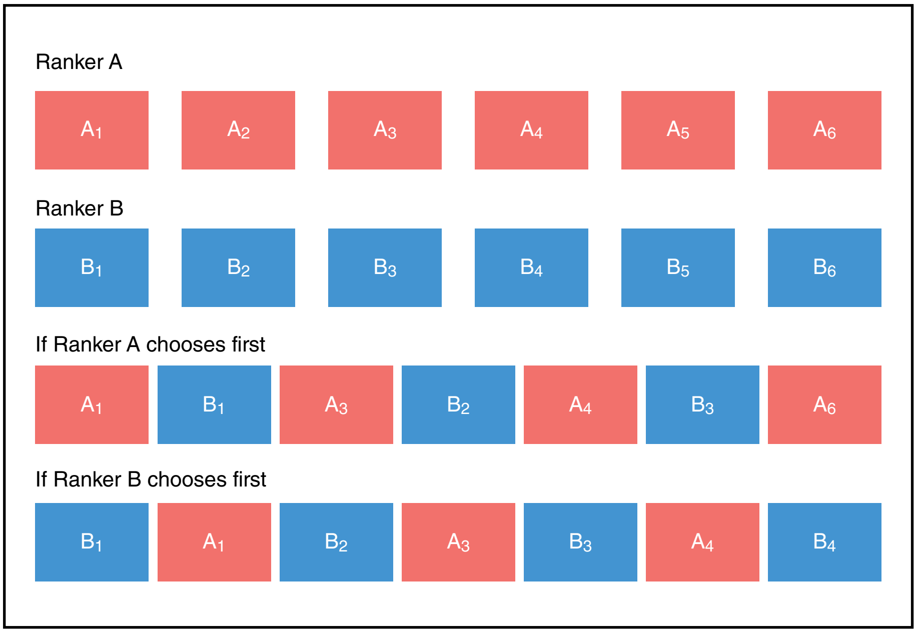

Figure 7.9 Interleaving method on a video ranking page [4].

choose players in a casual sports game. The alternative drafting approach is depicted in Figure 7.9.

After clarifying the specific evaluation process of the Interleaving method, it is necessary to verify whether this method can replace the traditional A/B test and whether it will produce wrong results. Netflix has verified the Interleaving method from two aspects – sensitivity and correctness.

7.5.3 Sensitivity Comparison of Interleaving and Traditional A/B Testing

Netflix's sensitivity experiments measured the sample population size required with the Interleaving method to achieve the same power as traditional A/B testing. Since the resources of online testing are often limited, experimenters always hope to use fewer online resources (for example, sample size, experiment period, and so on) for fast model evaluation.

Figure 7.10 shows the experimental results of the sensitivity comparison. The horizontal axis is the number of samples involved in the experiment, and the vertical axis is the p-value. It can be seen that the Interleaving method uses 10^3 samples to determine whether algorithm A is better than B, while the traditional A/B test requires 10^5 samples to reduce the p-value to below 5%. It shows the Interleaving method only needs 1% of the user sample to determine which algorithm is better. This means that 100 sets of Interleaving experiments can be done with the same level of resources needed by a single traditional A/B test, which undoubtedly greatly enhances the capacity of online testing.

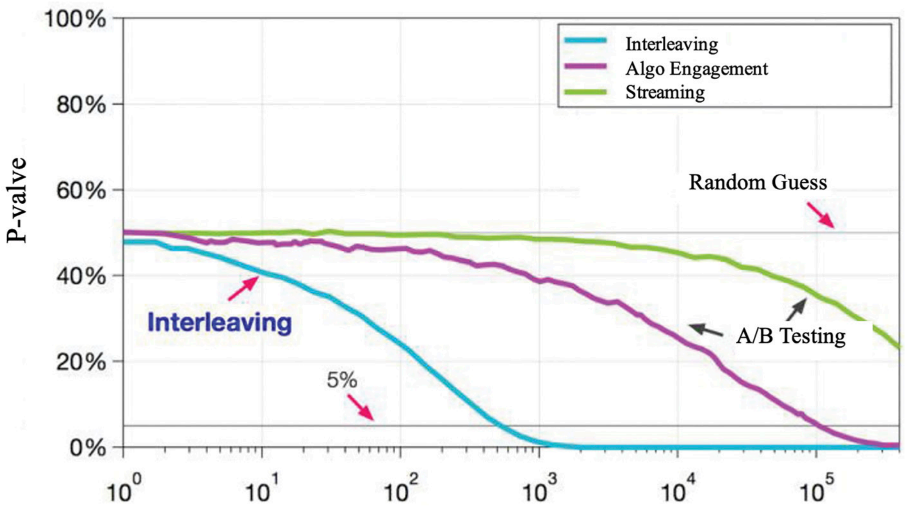

Figure 7.10 Sensitivity test for Interleaving and traditional A/B Testing [4].

7.5.4 Correlation of Metrics in Interleaving and A/B Testing

In addition to sensitivity, the result consistency between Interleaving method and A/B testing is the other key to determine the feasibility of Interleaving.

Figure 7.11 shows the correlation between the metrics from the Interleaving method and the A/B testing. Each data point represents a recommendation model. There is a very strong correlation between the Interleaving metric and the A/B test evaluation metric, which demonstrates that the algorithm that wins in the Interleaving experiment is also very likely to win in the subsequent A/B test.

It should be noted that although the correlation between the two test metrics is extremely strong – the pages displayed in the experiment of the Interleaving method are not actual production pages generated by algorithm A or algorithm B alone. So Interleaving can't totally replace A/B testing in order to measure the actual impact of the new algorithm. A/B testing is still the most authoritative testing method to get a comprehensive evaluation of model performance.

7.5.5 Advantages and Disadvantages of the Interleaving Method

The advantages of the Interleaving method are that it requires fewer samples, the test speed is fast, and the results are not significantly different from traditional A/B tests. However, readers should be clear that the Interleaving method also has certain limitations. The limitations are mainly in the following two aspects:

(1) The engineering framework is more complex than traditional A/B testing. The experimental logic and business logic of the Interleaving method are convoluted, so the business logic may be affected. In order to implement the Interleaving

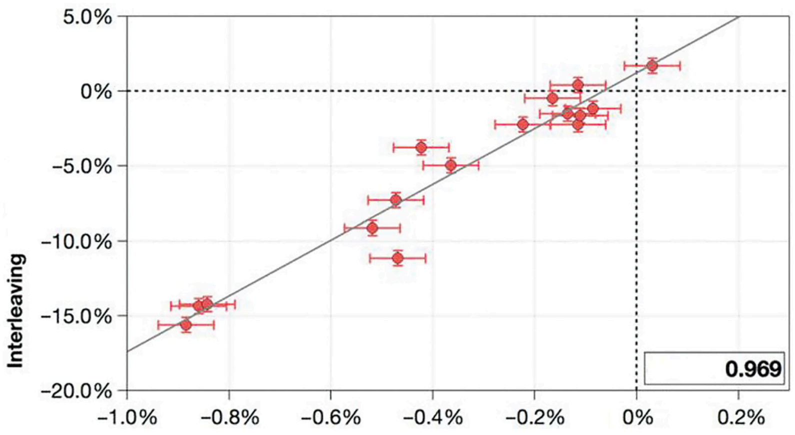

Figure 7.11 The correlation of Interleaving measurement with the A/B testing metric [4].

method, it is necessary to add a large number of auxiliary data identifiers to the entire data flow, which is a difficulty in engineering implementation.

(2) The Interleaving method is only a relative measurement of the user's response to the recommendation results, and cannot obtain the true performance of an algorithm. If you want to know how much one algorithm can improve the business KPIs, it is impossible to draw conclusions using the Interleaving method. To this end, Netflix designed a two-stage experimental structure of Interleaving + A/B testing to improve the entire online testing framework.

7.6 Recommender Systems Evaluation Architecture

This chapter introduces the main evaluation methods and metrics of recommender systems. These model evaluation methods are not independent. A mature evaluation architecture should comprehensively consider the evaluation efficiency and correctness. It needs to use fewer resources to quickly screen out models with better performance. This section systematically discusses how to build a mature recommender system testing and evaluation architecture using the introduced evaluation methods.

As Section 7.3 discusses, for a company, the most fair and reasonable evaluation method is to conduct online A/B testing to evaluate whether the model can better achieve the business goals of the company or the team.

In this case, why can't any model improvement be tested? The reason is given in Section 7.5 while introducing the Interleaving method. This is because online A/B testing takes up lots of online traffic resources and may also negatively impact user experience. However, the limited online testing bandwidths are far from satisfying the needs of model iteration and development. In addition, online testing often needs to last for several days or even weeks, which will greatly slow down the model iteration cycle.

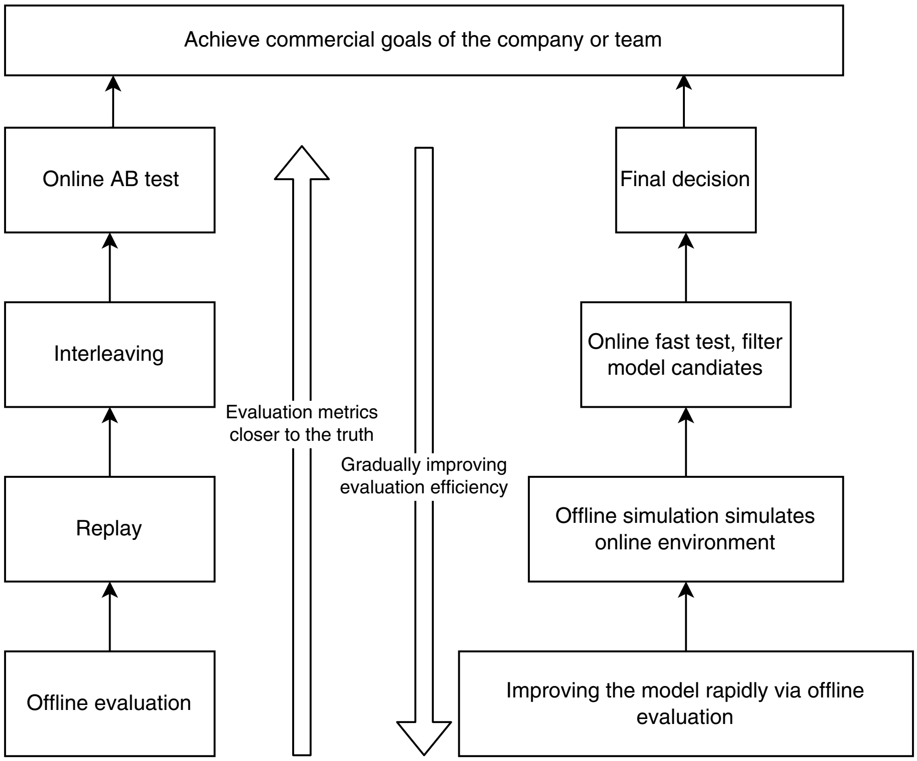
Achieve commercial goals of the company or team
Online AB test
Interleaving
Replay
Offline evaluation
Evaluation metrics closer to the truth
Gradually improving evaluation efficiency
Final decision
Online fast test, filter model candiates
Offline simulation simulates online environment
Improving the model rapidly via offline evaluation

Figure 7.12 Recommender system evaluation architecture.

Because of the limitations of online testing, offline testing has become the next best choice for model evaluation. Offline testing can use nearly unlimited offline computing resources to quickly obtain evaluation results, thereby quickly achieving iterative optimization of the model.

Between online A/B testing and traditional offline testing, there are evaluation methods such as Replay and Interleaving. The Replay method can simulate the online test process in the offline environment to the greatest extent, and the Interleaving method can establish a fast online test environment. This multilevel evaluation and testing method together constitute a complete recommender system evaluation architecture as shown in Figure 7.12, achieving a balance between evaluation efficiency and correctness.

In the schematic diagram of the evaluation architecture shown in Figure 7.12, the left side shows different evaluation methods, and the right side is the pyramid-shaped model screening process. It can be seen that the lower the level, the more models need to be screened, and the more improvement ideas need to be verified. Due to the huge number of possibilities, evaluation efficiency has become a more critical consideration, and the requirements for evaluation correctness are not so strict. At this time, a more efficient offline evaluation method should be used.

As candidate models are screened out layer by layer, the closer it is to the stage of final launch, the stricter the evaluation method's requirements for evaluating correctness should be. Before the model is officially launched, the final model evaluation should be done with the A/B test that is closest to the real production experience. After the most convincing online measurements are produced, the final model can be launched, and the iterative process of model improvement can be completed.

References

[1] Lihong Li, et al. Unbiased offline evaluation of contextual-bandit-based news article Recommender algorithms. Proceedings of the 4th ACM International Conference on Web Search and Data Mining, Hong Kong, China, February 9–12, 2011.
[2] Diane Tang, et al. Overlapping experiment infrastructure: More, better, faster experimentation. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25–28, 2010.
[3] Filip Radlinski, Nick Craswell. Optimized interleaving for online retrieval evaluation. Proceedings of the 6th ACM International Conference on Web Search and Data Mining, Rome, Italy, February 4–8, 2013.
[4] Joshua Parks, et al. Innovating Faster on Personalization Algorithms at Netflix Using Interleaving. Netflix Technology Blog. 2017. https://netflixtechblog.com/interleaving-in-online-experiments-at-netflix-a04ee392ec55