Common pitfalls in training and evaluating recommender systems

Hung-Hsuan Chen[†], Chu-An Chung[‡], Hsin-Chien Huang[‡], Wen Tsui[‡]
[†]Computer Science and Information Engineering, National Central University
[‡]Computational Intelligence Technology Center, Industrial Technology Research Institute
hhchen@ncu.edu.tw, {JudyChung, hchuang, arvin}@itri.org.tw

ABSTRACT

This paper formally presents four common pitfalls in training and evaluating recommendation algorithms for information systems. Specifically, we show that it could be problematic to separate the server logs into training and test data for model generation and model evaluation if the training and the test data are selected improperly. In addition, we show that click through rate – a common metric to measure and compare the performance of different recommendation algorithms – may not be a good measurement of profitability – the income a recommendation module brings to a website. Moreover, we demonstrate that evaluating recommendation revenue may not be a straightforward task as it first looks. Unfortunately, these pitfalls appeared in many previous studies on recommender systems and information systems. We explicitly explain these problems and propose methods to address them. We conducted experiments to support our claims. Finally, we review previous papers and competitions that may suffer from these problems.

1. INTRODUCTION

A recommender system suggests items that may interest a user. Recommender systems are mostly adopted by the E-Commerce (EC) providers, such as Amazon [19] and Walmart [14]. Researchers have proposed many recommendation algorithms from various perspectives, such as leveraging on the text similarity between products, utilizing users' clicking or purchasing behaviors, or a combination of both. These methods include Collaborative Filtering, Matrix Factorization, language modeling, and their variations and mixed versions [16; 19; 21; 22; 24].

To evaluate the proposed recommendation algorithms, researchers have applied or invented assorted metrics, such as the click through rate (CTR), the Mean Absolute Error (MAE), Normalized Mean Absolute Error (NMAE), and the Root Mean Square Error (RMSE) between the predicted user rating and the true user rating, the Discounted Cumulative Gain (DCG) and Kendall's Tau of the recommended item list, and the traditional Information Retrieval metrics like Precision and Recall [14].

Recommender systems can be applied to various application domains, such as movie recommendation [7], music recommendation [11], collaborator recommendation [8], expert recommendation [9], etc. Recently, industrial companies and research labs, such as Criteo, Netflix, and YOOCHOOSE, have organized recommendation competitions that attract thousands of participants. Typically, the organizer provides the datasets for training and locks away the test data for the final evaluation [6; 7; 15].

Instead of proposing another recommendation algorithm or evaluation metric, this paper emphasizes on four common pitfalls of developing and evaluating recommendation modules for information systems. Probably influenced by the machine learning, data mining, and information retrieval fields, the researchers of recommender systems usually split the available dataset (e.g., the logs of clickstreams) into the training and the test data, which are used to generate the recommendation model and evaluate the performance of the model respectively. Although such a procedure works well in many machine learning studies and applications, applying this procedure to recommender systems could be problematic, as we will explain later in this paper. In addition, the typical evaluation metrics – click through rate – may be problematic if used carelessly. Unfortunately, many of these pitfalls appear in previous research papers and competitions, as we will illustrate later. We discuss these issues and propose possible ways to fix or bypass them in this paper.

The rest of the paper is organized as follows. In Section 2, we present the typical approach to separate the training and the test data of the studies on recommender systems. We discuss two issues of generating and evaluating the recommendation methods in Section 3 and Section 4. In Section 5 and Section 6, we show two issues regarding click through rate and recommendation revenue. In Section 7, we review previous works, including (1) the typical recommendation metrics and (2) previous competitions and publications that fall into several pitfalls illustrated in this paper. Finally, we discuss the discoveries and address future work in Section 8.

2. A TYPICAL PROCEDURE OF PREPARING TRAINING AND TEST DATASETS

Figure 1 shows a possible procedure of generating the training and the test data when studying recommendation modules of an information system, e.g., an EC website. Initially, an EC website probably had no recommendation module. At a certain time, the engineers of the website decided to include a recommendation model. To train the model, the engineers used the available logs to extract (x_i, y_i) as the training instance. Here, $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,\ell})$ is the context features ($i = 1, 2, \ldots, n$; ℓ: the number of features; n: the number of training instances). The context features could be, for example, the user's gender, location, education,

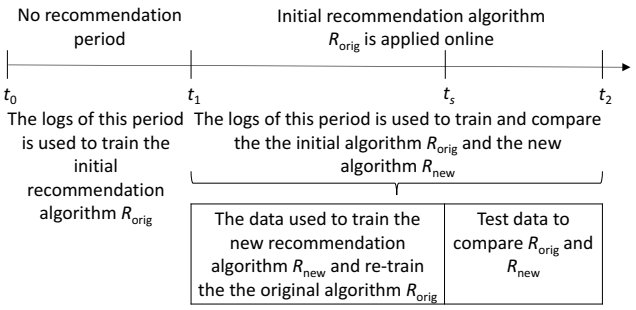

Figure 1: An illustration of a possible procedure to generate the training and the test data for developing and evaluating recommender systems. The initial online recommendation algorithm model R_{orig} is trained based on users' behaviors in the "no recommendation" period (from t_0 to t_1). To train a new model R_{new} and test whether the new recommendation method is better than R_{orig}, we split the log data between t_1 and t_2 into training and test data, as many research papers and competitions did. Both R_{orig} and R_{new} are trained based on the training data (the logs between t_1 and t_s) and compared based on the test data (the logs between t_s and t_2) using the specified metric (e.g., click through rate or order rate).

annual salary, the current browsing item's category, price, color, the current date, the current day of the week, etc. The y_i is the corresponding item that should be recommended to the user under the context features x_i. Since the website has no recommendation module initially, y_i could only be inferred based on intuitions, e.g., a user's next clicked or purchased item given x_i. During the test (recommendation) phase, the recommendation module utilizes the given context features as clues to output the products that may interest the users.

For example, if the engineers believe that a good recommender system should recommend products that are popular and related to the current browsing item, they may propose CategoryTP – recommending the top popular (e.g., most viewed) items the in the category c, the current browsing products' category. In this case, the context feature list contains only one feature $x_{i,1}$ whose value is c. To obtain the top popular items of each category, the view count of each product during the "no recommendation period" (see Figure 1) is calculated.

Suppose at another time, the engineers of the EC website proposed another recommendation algorithm R_{new}. To compare R_{new} with the original one R_{orig}, the engineers split the logs between t_1 and t_2 (the period where the original recommendation module was employed) into the training data (logs between t_1 and t_s) and the test data (logs between t_s and t_2). The engineers trained R_{new} the new model and re-trained R_{orig} the original model based on the training data, and compared their performance based on the test data using the specified metrics, such as the click through rate, the recommendation order rate, or the recommendation revenue.

Many research papers and competitions applied this procedure or similar procedures to generate the training and the test data, as will be illustrated in Section 7. Unfortunately, such a training and evaluation procedure is problematic, as demonstrated in the following sections.

Table 1: The percentage of the clicks resulted from the in-page direct links. We hide the actual numbers of clicks due to business sensitivity.

Day	Day 1	Day 2
Direct link Percentage	19.3150%	21.2812%

3. ISSUE 1: TRAINED MODEL COULD BE BIASED TOWARD HIGHLY REACHABLE PRODUCTS

This section shows that training a recommendation module based on the logs of the clickstreams may be problematic, if the training data are poorly selected. We will start by explaining the problem and proposing possible ways to bypass the problem. We will conduct experiments to support our claim.

3.1 The core problem

Many studies or competitions employ the clickstreams as the training data for the proposed recommendation algorithm [23]. Specifically, given that a user's current context feature as $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,\ell})$ and a user's next visited product as p_j, several researchers suggested to treat the recommendation task as a machine learning task in which (x_i, p_j) is a positive training instance. To generate the negative training instances, one possible approach is randomly sample a product p_k ($p_k \neq p_j$) from all products and treat (x_i, p_k) as a negative example [12].

If we generate the training dataset by this manner, the distribution of the training data (x_i, p_j) is affected by unstable factors, such as the presentation of the pages. Specifically, assume that we apply a Markov Chain recommendation model in which each node represents a product and $w_{i,j}$ the weight of a directed edge from node i to node j represents the transition probability from product p_i to p_j based on the log of the clickstreams. If the product page of p_i contains manifest direct-links to the product p_j, many users are likely to click p_j after browsing p_i. Thus, the information $p_i \rightarrow p_j$ (or more formally, $(x_i = (x_{i,1} = p_i), p_j)$ as a positive training instance) may become a strong positive signal simply because it is extremely easy to reach p_j from p_i. Unfortunately, the links included in the product pages are sometimes decided arbitrarily by few persons, e.g., the marketing executives or the engineers. As a result, the linked products may be little relevant to the current browsing product. For example, EC websites may aggressively exhibit the sponsored or advertised products, as demonstrated in Figure 2, even though the sponsored or the advertised products may not be directly relevant to the browsing product. As a result, the recommendation algorithms are likely to output the highly reachable products, which is highly influenced by the layout of the product pages.

3.2 Selecting proper training data

Based on the discussion above, we can see that a product's reachability highly influences the distribution of the train-

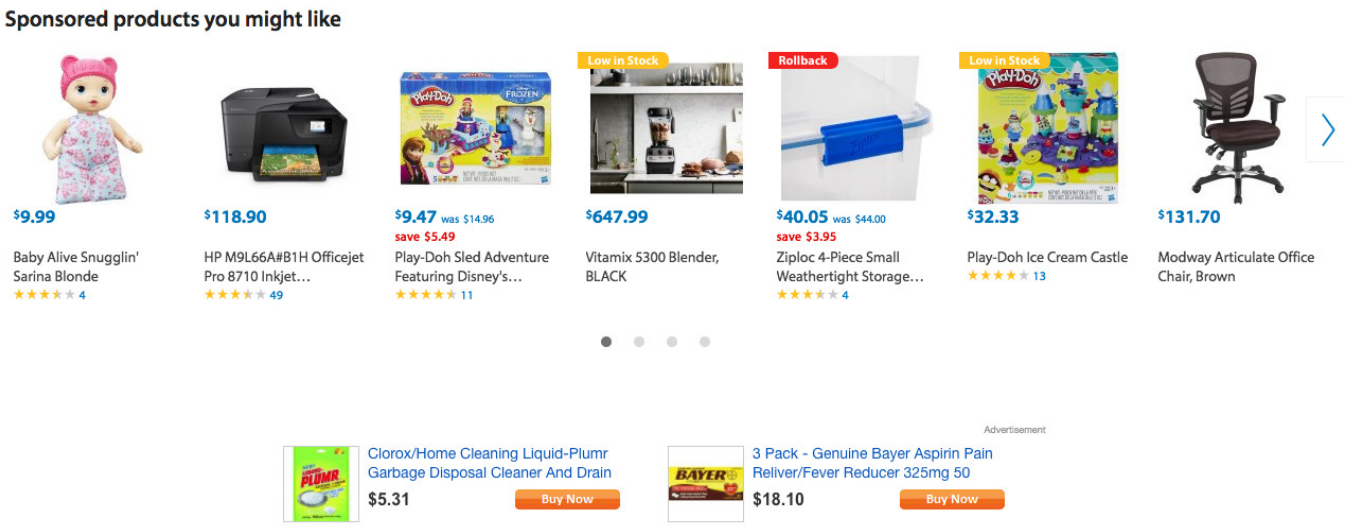

Figure 2: A snapshot of the product page on `https://www.walmart.com/`. The main product of this page is a prepaid smartphone. The sponsored products and the list of the advertised products on this page are of little relevance to a smart phone.

ing data. We illustrate two examples here. First, if we use the Markov Chain model as the training algorithm, the learned rules are highly influenced by the "layout" of the product pages (e.g., which product pages have direct links to which other product pages). Second, if we use popularity-based methods (e.g., obtaining the category of the current browsing product and recommending the top viewed products of the category), the recommendation algorithms are again likely to recommend the highly reachable products, such as the products that have many incoming links. Both cases are undesirable, because once we change the link targets, the distribution of the training data could be very different, which may lead to a different set of learned rules.

We analyzed the logs on two continuous days of a large EC website in Southeast Asia[1]. We found that about 20% of user clicks are resulted from the in-page direct links, as demonstrated in Table 1. This suggests that by rearranging the layout of the pages or the link targets in the pages, approximately 1/5 of the positive training instances are likely to be different.

To neutralize the effect that the highly reachable products are more likely to be treated as the positive training instances, we should weaken the weights of the positive instances in which next clicked product is highly reachable. In other words, if many product pages have direct links to the product p_j, we should assign a small weight to the positive training instance (x_i, p_j). Another possible method is to include the positive training instance (x_i, p_j) only when such a behavior is *spontaneous*, i.e., such a click was not influenced by the layout of the page. If we push the idea to the limit, we may include the information that p_i leads to p_j only when 1) the page of p_i contains no direct links to p_j, and 2) the p_j is not included in the recommendation list of the page of p_i. Thus, the positive signal that $p_i \to p_j$ is less likely to be affected by the layouts of the pages.

[1]Due to business sensitivity, we are not allowed to share the name of the company.

3.3 Experiment

To support the above claims, we simulated the typical method to generate the training data and the test data. This section presents the detailed settings, which is very close to what we introduced in Section 2. We present and discuss the results at the end of this section.

3.3.1 Experiment setting

We simulated an EC website that contains $1,000$ products $p_0, p_1, \ldots, p_{999}$. Each product is randomly assigned to one out of the ten product categories $c_0, c_1, \ldots, c_9$. We generated the similarity score between every pair of the products by the following rule: if two products p_i and p_j belong to different product categories, $s(p_i, p_j)$ the similarity score between them is sampled from a uniform distribution between 0.01 and 0.35. Otherwise, we sample $s(p_i, p_j)$ from a uniform distribution between 0.3 and 0.9.

The rest simulation follows the illustration of Figure 1. Initially (during t_0 and t_1), the website has no recommendation module. Each product page (p_i) contains direct links to 5 randomly selected products $(p_i^{(1)}, p_i^{(2)}, \ldots, p_i^{(5)})$, which we called "promoted products". We assume that when a user reads the page of product p_i, whether she/he will continue to view another product follows a Poisson distribution with a fixed λ. If she/he decides to continue, for 80% of the time the user would click on the promoted products and 20% of the time the user would visit a product p_j with probability proportional to $s(p_i, p_j)$ the similarity between p_i and p_j.

At t_1, the first recommendation module is online. We assume that if a user decides to stay on the site, then for 40% of the time she/he would click on one of the ten items returned by the online recommender system, 40% of the time the user would click on the promoted products, and for 20% of the time she/he would click the next item p_j proportional to the similarity score between p_i and p_j.

We generated 1,000,000 sessions that browse the products based on the above rules. We set the initial recommendation model to a simple approach CategoryTP (Categorical

Top Popular): when a user is viewing the product p_i whose product category is c_j, we recommend the top-10 popular products in the category c_j.

Now suppose we would like to try other recommendation algorithms. To compare the new algorithms with CategoryTP, we take the logs between t_1 and t_2 and separate the first 700,000 sessions as the training data and the last 300,000 sessions as the test data. We train each recommendation module (including the initial recommendation module CategoryTP) by two types training data: 1) train-all – all the 700,000 training logs, and 2) train-sel – if a user views p_i followed by p_j, the information is included in the training data only when p_j is neither included in the promoted products nor the recommendation list.

We compared the CategoryTP with the following methods: (1) TotalTP (Total Top Popular) : obtain the items that are mostly viewed during the training period and always recommend these items during the test period; (2) MC (Markov Chain): build a table to record the transition probability from one product to another during the training period and make recommendation in the test period based on the table, i.e., recommendation based on the most likely transferred product from the current browsing product; (3) ICF-U2I (Item-based Collaborative Filtering based on the User-to-Item matrix): during the training period, generate a user-to-item matrix in which the entry (i, j) represents the number of times a user i views a product j, and compute the cosine distance between every pair of the columns (which represents product features). During the test period, the algorithm recommends the columns that are most similar to the current browsing product; (4) ICF-I2I: (Item-based Collaborative Filtering based on the Item-to-Item matrix): during the training period, generate an item-to-item matrix in which the entry (i, j) represents the number of times item i and item j are viewed by the same user and compute the cosine distance between every pair of the columns (which represents product features). During the test period, the recommendation policy is the same as ICF-U2I, i.e., recommending the columns that are most similar to the current browsing product; (5) NMF-U2I: during the training period, generate a user-to-item matrix as explained in ICF-U2I and perform non-negative matrix factorization on the matrix to get the hidden vector of each item. Next, the algorithm calculates the cosine distance between every pair of these vectors. The recommendation policy during the test period is the same as ICF-U2I; (6) NMF-I2I: during the training period, generate an item-to-item matrix as explained in ICF-I2I and perform the non-negative matrix factorization on the matrix, as explained in NMF-U2I. The algorithm calculates the cosine distance and recommends items based on the distances during the test period, as explained in NMF-U2I.

3.3.2 Results

We show the average percentage of the promoted products appearing in the top-10 recommended items given the user is browsing a specified product. As demonstrated in Table 2, when using train-all (all the available training data) as the training data, several algorithms recommend many of the "promoted products". As a result, we seem to learn the "layout" of the product page (i.e., the direct links from p_i to p_j) instead of the intrinsic relatedness of between products. On the other hand, when we only utilize the train-sel (the data that match the rules specified in Section 3.2) as the training data, the ratio of the promoted products appearing in the recommendation list is much lower. In order not to let the layout of the product pages influence the learned rules, we probably should use the train-sel as the training data, or perhaps decrease the weights of the highly reachable products.

3.4 Lessons learned

The common wisdom that the clickstream represents a user could be problematic because the clickstreams are highly influenced by the reachability of the products and the layouts of the product pages. The items that occupy many spaces are more likely to be clicked and reached. As a result, training a recommender system based on the clickstreams are likely to learn (1) the "layout" of the pages, and (2) the recommendation rules of the online recommender system. Ideally, we should keep only the spontaneous clicks in the log to, at least partially, solve or bypass this issue.

4. ISSUE 2: THE ONLINE RECOMMENDATION ALGORITHM AFFECTS THE DISTRIBUTION OF THE TEST DATA

This section shows that evaluating the recommendation algorithms based on the logs of the clickstreams may be problematic if the test data are poorly selected. Like the previous section, we will start by explaining the problem and propose possible ways to bypass the problem. We will conduct experiments to support our claim.

4.1 The core problem

If we follow the procedure in Section 2 to generate the test data, the click through rate of the new proposed recommendation algorithm R_{new} is very likely to be worse than the online algorithm R_{orig}. The fundamental problem of such a setting is that, if the suggested product list L_{new} recommended by the new recommendation module R_{new} is very different from the online recommendation module's list L_{orig}, the online users have no chances to click on the products that appear only in L_{new} but not in L_{orig}. As a result, the recommendation modules that suggest products close to the online module tend to perform better.

4.2 Selecting proper test data

In several studies, researchers utilized the clickstreams as the ground truth for evaluating recommender systems, as we will demonstrate in Section 7. Specifically, given the current context feature x_i and the user's next visited product p_j, we treat a recommendation to be successful if the recommended list contains p_j. As we explained above, if a product p_k appears in L_{new} but not in L_{orig}, the product p_k is less likely to be clicked even though p_k might be a great recommendation given the context feature x_i.

If we use the entire clickstream logs as the ground truth for evaluation, the online recommendation algorithm is always favored, since the next prouct p_j is clicked probably because it is included in the recommendation list returned by the online recommendation module and naturally has a higher chance to be clicked. Instead of using the entire clickstreams, we probably should include the instance (x_i, p_j) to the test dataset only when the click happens spontaneously. Similar to the discussion in Section 3.2, if we push the concept to the limit, we should only include (x_i, p_j) to the test dataset

Table 2: The table shows the ratio of the number of the promoted products that appear in the recommendation list when using train-all or train-sel as the training data (each recommendation algorithm returns the top-10 results).

method	MC	CategoryTP	TotalTP	ICF-U2I	ICF-I2I	NMF-U2I	NMF-I2I
train-all	100.00%	1.48%	1.84%	93.22%	1.40%	1.48%	1.34%
train-sel	1.08%	0.86%	0.98%	14.46%	1.28%	1.32%	1.24%

if 1) p_j is not directly linked from the current browsing page, and 2) p_j is not included in the recommendation list given x_i as the input of the online recommendation algorithm.

4.3 Experiment

This section demonstrates that the online recommendation algorithm highly influences users' clicks.

4.3.1 Experiment setting

The entire simulation process is very similar to Section 3.3. Specifically, we first use the CategoryTP as the online recommendation algorithm and report the click through rate of various recommendation algorithms based on two test datasets: (1) test-all – the entire clickstream logs, and (2) test-sel – the clickstreams that match the conditions described in the last section. Next, we change the online recommendation algorithm to TotalTP and repeat the same experiment.

4.3.2 Results

Table 3 and Table 4 show the experimental results when the online recommendation algorithms are CategoryTP and TotalTP respectively. For each compared method in each table, we show four results: (1) training by train-all and evaluating by test-all; (2) training by train-all and evaluating by test-sel; (3) training by train-sel and evaluating by test-all; (4) training by train-sel and evaluating by test-sel.

If we use train-all for training and test-all for evaluation, the recommendation algorithm that is the same as the online recommendation algorithm always yields great results. For example, when we use CategoryTP as the online recommendation algorithm, the click through rates of (offline) CategoryTP and TotalTP are 37.61% and 5.41% respectively. However, when we switch the online algorithm to TotalTP, the click through rate of CategoryTP drops to 5.28%, and the click through rate of (offline) TotalTP increases to 40.80%. This demonstrates the issue we discussed in Section 4.1 – the online recommendation algorithm and the ones similar to the online recommendation algorithm tend to outperform the others if the training and the test datasets are selected poorly. Since several previous studies simply use all the available test dataset for evaluation, the results are likely to bias toward the online algorithm. As a result, their reported results are questionable.

When we use train-all for training and test-sel for evaluation, the models probably partially learned the recommendation rules of the online recommendation algorithm. If the online algorithm is a mediocre algorithm (e.g., TotalTP), the proposed algorithms are likely to learn from bad examples. Thus, the models are probably not generic enough to perform well in the general cases. As demonstrated in Table 4, all the (train-all, test-sel) are worse than the (train-sel, test-sel) results given the same proposed recommendation algorithm. On the other hand, if the online recommendation algorithm mostly offers great recommendations, even a mediocre algorithm may learn from good examples (obtained from train-all). However, the good results do not result from the ability of the model itself, but from the good training data. As a result, the reported numbers may still be questionable.

When we use train-sel as the training data and test-all as the test data, the evaluation ground truth is affected by the online recommendation algorithm and the link structure between the product pages. As demonstrated, when using CategoryTP as the online algorithm, the click through rates of (offline) CategoryTP and TotalTP are 4.36% and 0.71% respectively. However, when using TotalTP as the online algorithm, the click through rate of CategoryTP drops to 0.86% and the (offline) TotalTP increases to 1.47%. Indeed, the online algorithm affects the distribution of the test dataset.

Finally, we believe that the proper training dataset and test dataset should be train-sel and test-sel respectively. In this case, the online training model does not affect the generation of the training data. Thus, the proposed methods will not learn from the examples that are affected by the online learning algorithm. Meanwhile, the online training algorithm does not affect the generation of the test data either. Thus, we may evaluate the proposed method based on an unbiased test dataset.

4.4 Lessons learned

Previous studies sometimes use all the available test data as the ground truth for evaluation. Unfortunately, such an evaluation process inevitably favors the algorithms that suggest products close to the online recommendation algorithm. We should carefully select the test dataset to perform a fairer evaluation.

5. ISSUE 3: CLICK THROUGH RATES ARE MEDIOCRE PROXY TO THE RECOMMENDATION REVENUES

5.1 Core problem

Several academic studies on recommender systems exploit the click through rate to compare different recommendation algorithms. This metric is user-centric, i.e., it measures a user's satisfaction about a recommendation. Click through rate is popular, probably because the industry hesitates to share or release revenue-related information. As a result, researchers mostly can only study users' feedback and satisfaction on a recommender system and hope that boosting user-centric measures (e.g., click through rate) will increase the business-centric measures (e.g., recommendation revenue). Unfortunately, such a surmise was not carefully validated.

Table 3: The click through rates when the online recommendation algorithm is CategoryTP.

	CategoryTP		TotalTP		MC		ICF-U2I		ICF-I2I		NMF-U2I		NMF-I2I	
	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel
test-all	37.61%	4.36%	5.41%	0.71%	58.89%	4.05%	46.88%	17.02%	28.37%	21.79%	16.54%	5.78%	8.78%	16.19%
test-sel	2.76%	2.75%	1.02%	1.03%	1.88%	2.86%	1.60%	3.03%	2.38%	2.66%	2.54%	1.74%	2.52%	2.73%

Table 4: The click through rates when the online recommendation algorithm is TotalTP.

	CategoryTP		TotalTP		MC		ICF-U2I		ICF-I2I		NMF-U2I		NMF-I2I	
	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel	train-all	train-sel
test-all	5.28%	0.86%	40.80%	1.47%	60.40%	1.22%	48.61%	16.40%	23.85%	10.77%	13.29%	0.80%	1.53%	4.17%
test-sel	2.66%	2.89%	0.93%	1.18%	1.01%	2.98%	1.04%	2.62%	1.00%	2.07%	1.08%	2.39%	1.12%	2.03%

Figure 3: The relationship between the click through rate and the recommendation revenue (from 2014/9/2 to 2015/9/13, each point represents one date). We exclude the tick marks because we are not allowed to report their exact values. The correlation of determination is only 0.089, suggesting that they have a weak positive relationship.

5.2 Experiment

5.2.1 *Experiment setting*

We selected two measures to represent the user-centric and the business-centric metrics: the click through rate and the recommendation revenue. The click through rate is the proportion that a recommendation is clicked. This metric is probably the most typical user-centric measure. The recommendation revenue is the income contributed by the recommendation module.

We collected a year-long log (2014/9/2 - 2015/9/13) from a large EC website in Southeast Asia. For each day, we calculated the click through rate and the recommendation revenue. Note that during the year of the study we launched and retired different recommendation algorithms (e.g., Matrix Factorization, Markov Chain, CategoryTP, a mixture of several methods, etc.) on different pages (e.g., the main page, the product category page, and the product page). As a result, we measure the click through rate and the recommendation revenue across many different settings.

5.2.2 *Experiment results*

Figure 3 presents the relationship between the click through rate of the recommendations and the recommendation revenue. The correlation of determination (R2) between the click through rate and recommendation revenue is only 0.089, suggesting that they have a very weak correlation. Therefore, it could be improper to simply pursuing click through rate and hope that increasing click through rate will boost the revenue.

5.3 Lessons learned

Based on the result, comparing different recommendation modules purely based on the user-centric metrics, such as the click through rate, may fail to capture the business owner's satisfaction. Unfortunately, studies on recommender systems mostly perform comparisons based on the user-centric metrics. As a result, even if a recommendation algorithm attracts many clicks, we cannot assure this algorithm will bring a large amount of revenue to the website.

6. ISSUE 4: EVALUATING RECOMMENDATION REVENUE IS NOT STRAIGHTFORWARD

EC companies build recommendation modules in the hope that these modules will discover users' purchasing intentions and eventually boost the revenue. It is reported that a large portion of an EC website's revenue comes from recommendation [20]. In this section, we show that evaluating recommendation revenue is not as straightforward as it first looks.

6.1 The core problem

We ask a more fundamental and probably more challeng-

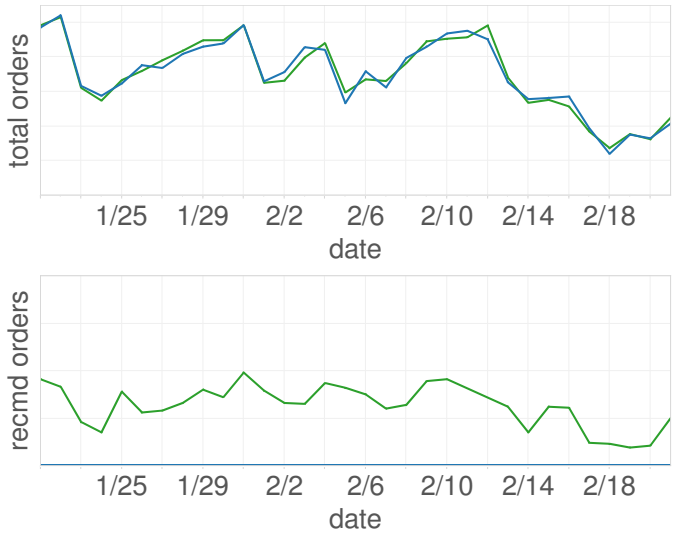

Figure 4: A comparison of the number of the total purchases in two cases: displaying the recommendation panel or not. Green line: the channel with a recommendation panel; blue line: the channel without a recommendation panel. Due to business sensitivity, the tick labels of the y-axes are removed. It appears that the recommendation panel brings little *extra* revenue, if any, to the channel.

ing question: does a recommendation module bring *extra* revenue (or *extra* orders) to an EC website? Although we can measure the number of purchases contributed by a recommendation module, we cannot tell whether these buyers would still purchase these (or possibly similar) items without the recommendation module. It is possible that the recommendation modules are served as a convenient tool for users to locate the desired items, but even without the recommendation module, the users can still discover these items through another user interface provided by the website.

6.2 Experiment

6.2.1 *Experiment setting*

To answer the question, we propose to conduct experiments based on A/B testing to control the appearance of the recommendation module. Specifically, we collaborated with a large EC website in Southeast Asia. We directed 5% of users to a channel that displays no recommendation on the product page (channel 1), and another 5% of users to a normal channel that displays recommendations as usual (channel 2). If the *total* purchases of channel 1 is very similar to the total purchases of channel 2, then the recommendation module brings no *extra* purchases to the website. In this case, the recommendation module probably simply provides another way for the users to discover the items of their interests.

6.2.2 *Experiment results*

Figure 4 shows the total purchases and the recommended purchases of the two channels. We use blue line to display users' orders in channel 1 (the no recommendation channel) and the green line for channel 2 (the normal channel). The lower sub-figure of Figure 4 shows that users indeed purchase recommended items. This seems to suggest that recommendation is helpful in increasing the number of sales. However, if we compare the total number of purchases of each channel (the upper sub-figure of Figure 4), we see no obvious benefit of having the recommendation panel. It appears that, for most of the times, the users who purchases recommended items still purchase items even without the recommendation module.

6.3 Lessons learned

Based on the result, we suspect that, although a recommendation module may help users quickly discover their needs, these users, even without the recommendations, may still be able to locate the desired products by other processes, e.g., by querying through the search bar, or by navigating through the hierarchical table of contents. Thus, it is not clear whether a recommendation module brings extra purchases, or simply re-direct users from other purchasing processes to recommendation. In an extreme case, an EC company can fill in the entire page with recommendations and claim that nearly 100% of their revenue comes directly from recommendations. Apparently, such a claim is misleading. To proper evaluate the extra revenue contributed by a recommender system, we still need to leverage on A/B testing. Unfortunately, it is very difficult for the researchers in academia to perform A/B testing in practice.

7. RELATED WORKS

7.1 Common metrics to evaluate recommender systems

Most studies on recommender systems compare different algorithms based on user-centric metrics, which can be categorized into the following types: accuracy-based, diversity-based, and surprisal-based.

Typically, we would like a recommendation module to accurately predict a user's preference on items. Thus, accuracy is a natural choice to measure recommendation modules. A simple way to measure accuracy is click through rate – in what percentage a user clicks a recommendation [10]. However, such a metric may favor the algorithms that tend to recommend popular items, because recommendation accuracy usually declines towards the long tail [25]. In addition to accuracy, researchers have found that diversity plays an important role in improving users' satisfaction about recommendation [5]. To quantify "diversity", one can measure the average dissimilarity between all pairs of recommended items based on these items' attributes, such as their brands, prices, or taxonomy. Diversity and accuracy are usually a trade-off. One can easily increase diversity by recommending unrelated items, but this usually sacrifices the accuracy. Surprisal, sometimes known as novelty, is a type of user-centric metric that is relevant to diversity. Previous studies have inconsistent definitions about surprisal. As a result, the terms "diversity", "surprisal", "serendipity", and "coverage" are sometimes interchangeable. Here, we define surprisal as the unexpectedness of a recommendation. Thus, one can define the surprisal as the inverse of an item's popularity. Such an idea is illustrated in [27] with a simple modification.

Several studies aimed to optimize a combination of the above metrics because very often we can expect a tradeoff between these metrics. Instead of proposing another metric or trying

to maximize these metrics, we show that many studies probably incorrectly reported these metrics. We also show that user-centric metrics may be inconsistent with the business-centric metrics, such as recommendation revenue.

There are studies aimed at unbiased offline evaluation for recommender systems based on contextual-bandit [18; 17]. These methods mostly assume the researchers can control the online recommendation module to explore the uncertain cases. Unfortunately, most researchers have no access to a live large-scale information system. They mostly rely on the datasets released by the large companies.

7.2 Reviewing Previous Competitions and Publications

We review previous competitions and publications that may fall into some of these pitfalls.

RecSys Challenge 2015 [6] offered the clickstreams of sessions from a big retailer which utilizes the recommender system service provided by YOOCHOOSE. Some of these sessions include the purchase events. The goal of the challenge is to predict the purchased item (if any) given the clickstream of a new session. Such a dataset is indeed valuable resource for the researchers in academia. However, using the clickstreams as the training data may generate the models that tend to recommend the highly reachable items, as discussed in Section 3. Additionally, the challenge utilized users' logs to perform offline evaluation, which may also be problematic, as explained in Section 4. The similar problems appear in several challenges, such as the RecSys Challenge 2016 [26], the Display Advertising Challenge provided by Criteo Labs in 2014 [3], the Click-Through Prediction Challenges provided by Avazu in 2015 [2] and by Outbrain in 2017 [4]. The studies that utilized these datasets for model generation and evaluation may also suffer from similar problems. Recently, some competitions started to use (or partially use) the online events for evaluation. For example, the RecSys Challenge 2017 consists of two stages – the offline and the online phase. During the online evaluation phase, the recommendations proposed by each team are rolled out to the live system. As a result, the real users have chances to interact with the recommended items [1].

Many studies on recommender systems aimed at predicting users' ratings to items or increase the click through rate. However, clicking (browsing) and buying could be very different behaviors. As shown in [13], this two types of behaviors can be classified by a supervised learner. Thus, simply pursuing the click through rate may not necessarily increase the business runner's revenue. For example, given a list of recommended items, a user may be encouraged to continue browsing these items instead of purchasing. Unfortunately, since the revenue related information is usually business sensitive, the business runners may not want to share such information with outsiders.

It is reported that 35% of Amazon's product sales are from recommendation [20]. However, how to derive such a number was unspecified. If, for example, an EC website fills most of the pages with recommendations, it is not surprising that most of the clicks and the purchases are directly or indirectly from the recommendations. To ensure the (extra) purchases or the (extra) revenue coming from a recommendation module, we believe that one of the most reliable tool is A/B testing. Unfortunately, applying A/B testing requires to have a large platform with many users. This is not always available for researchers, especially those in universities.

8. DISCUSSION AND FUTURE WORK

This paper shows four pitfalls that may occur in several studies and competitions of recommender systems. Specifically, the first two issues are due to the biased data collection of the training and the test datasets. These two pitfalls should be concerned not only by the researchers and practitioners of the recommender systems but also the data scientists in general. However, we mainly focus on the field of recommender systems in this paper because, unlike some machine learning problems in which the distribution of the training and the test datasets are affected only to a small extent by the data collecting approach, the distribution of the training and test dataset of recommender systems are highly influenced by the way they are collected. The third issue is regarding the proper selection of the evaluation metrics. Again, every data scientist should aware of the issue, but such a problem is less discussed in the field of recommender systems due to its nature – the revenue related information is usually business sensitive and protected. Finally, the fourth issue is even less addressed: even if the recommender systems indeed bring purchases, the purchases may not be the *extra* purchases. As a result, the recommender systems may improve customers' user experience, but may not bring immediate extra revenue to a business runner.

For future work, we would like to build an open platform in which researchers may register and plug their recommendation algorithm onto the platform. The platform redirects traffic into different registered algorithms in the backend. Thus, every recommendation algorithm is served online. The researchers do not need to worry about the offline evaluation issues. Such a platform may, at least partially, address the second and the fourth issue discussed in this paper. We are currently developing such a system and negotiating with several EC companies to try the system. We hope to open the system as an arena so that researchers and practitioners may compare their recommendation algorithms in a more realistic environment.

9. ACKNOWLEDGEMENTS

We appreciate partial financial support from the Ministry of Science Technology (MOST 105-2218-E-008-015) and the Industrial Technology Research Institute (ITRI 106-W100-21A2). We are grateful to the National Center for High-performance Computing for computer time and facilities.

10. REFERENCES

[1] ACM RecSys Challenge 2017. `http://2017.recsyschallenge.com/`. Accessed: 2017-07-14.

[2] Click-through rate prediction. `https://www.kaggle.com/c/avazu-ctr-prediction`. Accessed: 2017-07-14.

[3] Display advertising challenge. `https://www.kaggle.com/c/criteo-display-ad-challenge`. Accessed: 2017-07-14.

[4] Outbrain click prediction. `https://www.kaggle.com/c/outbrain-click-prediction`. Accessed: 2017-07-14.

[5] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*, pages 5–14. ACM, 2009.

[6] D. Ben-Shimon, A. Tsikinovsky, M. Friedmann, B. Shapira, L. Rokach, and J. Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 357–358. ACM, 2015.

[7] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[8] H.-H. Chen, L. Gou, X. Zhang, and C. L. Giles. CollabSeer: a search engine for collaboration discovery. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 231–240. ACM, 2011.

[9] H.-H. Chen, I. Ororbia, G. Alexander, and C. L. Giles. ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries. *arXiv preprint arXiv:1511.02058*, 2015.

[10] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[11] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Advances in neural information processing systems*, pages 385–392, 2008.

[12] Y. Goldberg and O. Levy. word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[13] Q. Guo and E. Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 130–137. ACM, 2010.

[14] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[15] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin. Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 43–50. ACM, 2016.

[16] Y. Koren, R. Bell, C. Volinsky, et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[17] L. Li, S. Chen, J. Kleban, and A. Gupta. Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International Conference on World Wide Web*, pages 929–934. ACM, 2015.

[18] L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306. ACM, 2011.

[19] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[20] I. MacKenzie. How retailers can keep up with consumers. `http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers`. Accessed: 2017-07-14.

[21] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[22] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[23] P. Romov and E. Sokolov. Recsys challenge 2015: ensemble learning with categorical features. In *Proceedings of the 2015 International ACM Recommender Systems Challenge*, page 1. ACM, 2015.

[24] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[25] H. Steck. Item popularity and recommendation accuracy. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 125–132. ACM, 2011.

[26] W. Xiao, X. Xu, K. Liang, J. Mao, and J. Wang. Job recommendation with hawkes process: an effective solution for recsys challenge 2016. In *Proceedings of the Recommender Systems Challenge*, page 11. ACM, 2016.

[27] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.