Generalized Negative Sampling for Implicit Feedback in Recommendation

Yuki Yamanaka
Kyoto University
Kyoto, Japan
yuukiyamanaka13@db.soc.i.kyoto-u.ac.jp

Kazunari Sugiyama
Kyoto University
Kyoto, Japan
kaz.sugiyama@i.kyoto-u.ac.jp

ABSTRACT

In a typical model-based collaborative filtering with implicit feedback, negative sampling is getting more and more popular to obtain negative labeled inputs from massive unobserved data. However, this approach wrongly samples false negatives, resulting in unacceptable recommender model. In this work, we first identify a situation where false negatives are problematic and estimate their impact on a model accuracy. Then, we take our negative sampling as a classification task and demonstrate that a recommender model and a negative sampling method actually share the same goal: identification of false negatives from other unobserved data. We also estimate the actual upper bound of the accuracy improvements with a feasible negative sampling. Lastly, we propose a generalized negative sampling that can alleviate the impact of false negatives by introducing two robustness against false negatives: self-sampling and dynamic sub-sampling. In user-item interaction matrix, experimental results on publicly available datasets show that our approach outperforms some state-of-the-arts with statistical significance.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Implicit feedback, Collaborative filtering, Matrix factorization, Negative sampling, False negative

ACM Reference Format:
Yuki Yamanaka and Kazunari Sugiyama. 2021. Generalized Negative Sampling for Implicit Feedback in Recommendation. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT '21), December 14–17, 2021, ESSENDON, VIC, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3486622.3493998

1 INTRODUCTION

Recent works on recommender systems have often employed implicit feedback such as each user's clicks. However, the implicit feedbacks do not contain explicit information about whether each user actually prefers a target item or not. Thus, labeling framework on these data is widely used to train a recommender model [9]. Generally, if a user u selected an item i, "1" is labeled to the input (u, i), otherwise, "0" is labeled.

This labeling framework assume that a user likes items if he interacts with them and dislike them if he does not. However, this assumption poses a major concern: false negatives, in which some items a user does not select are potentially positive for the user. In the context of offline evaluation, the test data would be wrongly labeled 0 instead of 1 if we were to use the aforementioned labeling framework. As recommender models would be trained in a way that all user-item pairs except for the training data would be labeled 0, the model would output 0 even for the test data. However, actually, the model is expected to output 1 for them. Therefore, test data could be regarded as false negatives in offline evaluation.

When training a model, it is irrelevant to employ the above labeling strategy, in which user-item pairs in training data are labeled 1 and all others 0. It clearly results in over-fit to train data as the model is trained to output 0 even for user-item pairs in test data. Therefore, a sampling technique is widely used to select some user-item pairs from all data except training data and labels 0 (*i.e.*, negative) to them, often referred to as *negative sampling*. To address inefficient training and ineffective problems with random sampling, many approaches have been proposed. (*e.g.*, rank-biased-based [16, 22], adversarial-based [1, 6, 14], and external knowledge-based [4, 20]). However, they overlook the problem with false negatives. Recent negative sampling techniques have focused on *correctness* of sampled items, excluding false negatives from sampled data [4, 5, 20]. However, they need more specific information as well as the user-items interaction matrix, making it difficult to adapt them to other scenarios. To solve this, we propose a generalized negative sampling which can adapt to any recommendation scenarios. Our main contributions can be summarized as follows:

- We analyze the impact of false negatives on a model accuracy. In our experiments, we observe that false negatives significantly give negative effect on a specific type of negative sampling. To the best of our knowledge, we are the first to experimentally show the impact of this issue.
- We carefully analyze negative sampling and show the upper bound of improvement in model accuracy that can be achieved by removing the effect of false negatives. As far as we know, our work is also the first to theoretically and experimentally show the upper bound.
- We propose a generalized negative sampling framework which can alleviate the negative impact of false negatives and can approach to the upper bound. Experimental results on three publicly available datasets show that our proposed method outperforms some state-of-the-arts.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WI-IAT '21, December 14–17, 2021, ESSENDON, VIC, Australia
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9115-3/21/12…$15.00
https://doi.org/10.1145/3486622.3493998

2 ANALYSIS ON FALSE NEGATIVE

2.1 Preliminaries

Notations and Problem Formulation. Before we discuss the problem with false negatives, we first introduce notations and problem formulation to help understand the subsequent discussions. This work considers traditional recommendation scenario: recommendation only using user-item interaction matrix. Let $S \in U \times I$ be a set of positive (*i.e.*, observed) interactions in training data where U and I are a set of users and items, respectively. Let S' and S^- be a set of positive interactions in test data and a set of unobserved interactions, respectively. Let I_u and I'_u be a set of positive items for a user $u \in U$ in training and test data, respectively. The task of recommendation is to find the most relevant ranking r of items for each user, which can be formulated by the following function:

$$r_\theta(i|u) : I \times U \rightarrow \{1, 2, ..., |I|\},$$

where r_θ is the rank of item $i \in I$ for a given user u. The ranking function is usually modeled by a predicted scoring function $\hat{y}_\theta(i|u)$, parameterized by a model θ. The link between the rank r_θ and the scoring function $\hat{y}$ can be defined as follows:

$$r_\theta(i|u) := |\{j : \hat{y}_\theta(j|u) \geq \hat{y}_\theta(i|u)\}|.$$

Furthermore, we can take the ranking as a binary classification, in which the model classifies each item i to "prefer" or "not prefer" for each user u. Using ranking information, the model D_θ can classify items into "prefer" (= 1) and "not prefer" (= 0):

$$D_\theta(i|u) := \begin{cases} 1 & (r_\theta(i|u) \leq |I_u \cup K|) \\ 0 & (otherwise), \end{cases} \tag{1}$$

where K is a predicted $|I'_u|$ because the model D_θ cannot know the actual $|I'_u|$. Then, the classification error of test data for each user u is defined as:

$$e'_\theta(u) = \frac{1}{|I'_u|} \sum_{i \in I'_u} (1 - D'_\theta(i|u)). \tag{2}$$

Thus, the overall classification error of test data is defined as:

$$e'_\theta = \frac{1}{|U|} \sum_{u \in U} e'_\theta(u). \tag{3}$$

Our work focuses on minimizing the error e'_θ in Equation (3).

Recommender Models. We address a group of recommender models which require intentional data labeling (*i.e.*, models which use negative sampling to train). For example, matrix factorization-based methods [12, 13, 15] and neural network-based methods [7, 21] are in our scope. We employ Biased MF [13] in our experiments as it is simple and robust for the traditional recommendation scenario [3, 18] compared with neural network-based models.

The Biased MF first represents users (u) and items (i) as latent factors p_u and q_i, respectively. Then preference score is computed by the function y defined as follows:

$$\hat{y}(i|u) = p_u \times q_i + \mu + b_u + b_i,$$

where b_u and b_i are bias terms for u and i, respectively. μ is a regularization term.

To optimize it, binary cross entropy (BCE) is often employed as an objective function, which compares predicted preference score $\hat{y}(i|u)$ with intentional label $y_{ui} \in \{0, 1\}$. If an interaction between u and i are observed, then $y_{ui} = 1$, otherwise 0:

$$l_{BCE} = \sum_{u, i \in S \cup S^-} y_{ui} \log \hat{y}(i|u) + (1 - y_{ui}) \log(1 - \hat{y}(i|u)). \tag{4}$$

It is natural that Equation (4) looks like Equation (2) as BCE is originally adapted to a classification task. Our experiments employ BCE as an objective function.

2.2 Negative Sampling

We need both positively and negatively labeled data as inputs to train a model. Generally, we want to use all observed and unobserved data as the positive and negative inputs, respectively. However, this is not practical in terms of computational complexity. To overcome this, many sampling methods have been proposed so far. If m negative items is sampled from unobserved data for each positive observed data, the number of negative items in model input is reduced to m from $|S^-|$. Among various sampling methods, the most popular one is random sampling [17], which uniformly samples m items from unobserved data.

In terms of false positives, we can argue that negative sampling is generally robust to false negatives. As discussed in Section 1, negative sampling can avoid some false negatives during training, while training with the whole data results in overfitting to training data. However, as most items quickly converge to values close to the label (*i.e.*, 0 or 1), random sampling cannot efficiently sample items which give large errors [5, 16]. As a result, the loss of sampled items quickly decreases, and the model accuracy is not further improved. In particular, the recommendation accuracy for higher top-K (*e.g.*, $K = 5, 10$) is insufficient.

To overcome this, rank-biased sampling approaches have been proposed [16, 22]. This framework selects a higher-ranked unobserved item $j \in I \setminus I_u$ scored by the current model with higher probability. The intuition behind this approach is to obtain a larger gradient by sampling true negatives with high prediction scores. Adaptive oversampling baysian personalized ranking (AOBPR) [16] samples unobserved interaction j as negative with the following distribution:

$$j \sim p(j|u) \propto r(j|u)^{-s},$$

where s is a polynomial degree. If $s \rightarrow 0$, $p(j|u)$ approximates to random sampling. If $s \rightarrow \infty$, the top-ranked item among all items always be sampled. As AOBPR needs to rank all items for each user, complexity $O(|U||I|log|I|)$ is required if naively calculated. If the number of items $|I|$ is large, this additional computation could be a significant problem. To compute more efficiently, dynamic negative sampling (DNS) [22] first samples a subset (candidates) $C \subset I$ with uniform distribution, and then chooses an instance $j \in C$ scored higher by the current model. Thus, DNS can reduce additional computational complexity to $O(|U||C|log|C|)$.

Rank-biased sampling shows promising performance in evaluation with some ranking metrics such as nDCG [10] and mean average precision (MAP). However, it overlooks the false negative issue, resulting in lower performance during training a model. In the initial stage of training, rank-biased sampling can correctly sample true negatives in unobserved interactions. On the other hand, it frequently samples false negatives in the latter stage as

the model gradually ranks false negatives higher (*i.e.*, test data) by training, resulting in lower recommendation accuracy.

However, we observe that DNS is superior to AOBPR not only in computational efficiency but also in model performance, indicating that DNS intrinsically has a robustness against false negatives. This is because the probability of sampling false negatives in the top of ranking sub-sampled items is generally lower than that in the top of ranking all items. Especially, DNS can control a robustness against false negatives by tuning the size of subset C. DNS is approximate to AOBPR if $C \to I$ while it is approximate to random sampling if $C \to 1$. This effect of sub-sampling is discussed in Section 2.4. We employ DNS as a representative rank-biased sampling.

2.3 The Impact of False Negatives

We first clarify the effect of false negatives by assuming an ideal sampling method that does not sample false negatives completely. Then we can estimate the impact of false negatives by comparing the accuracy of a model trained using this sampling with that using ordinal sampling, which possibly sample false negatives. We implement the ideal sampling, in which the sampling probability of false negatives is set to 0 by artificially using test data. Although this sampler examine the test data and propagate information on it to a model during training (data leakage), it does not use the test data as positive inputs during training.

Experimental Setup. Here, we tuned hyper-parameters for ideal sampling, in which we also use the same experimental settings in Section 5.

Results. Tables 1a and 1b compare Precision@10 and @20 when the model is trained using ordinary and ideal sampling in random sampling and DNS, respectively. At first glance, while ideal sampling gives slight improvement in random sampling (see Table 1a), it gives signifcant improvement in DNS (see Table 1b). In random sampling, as the probability of sampling a false negative is not high even in usual training, false negatives do not matter.

On the other hand, in rank-biased sampling, as the probability of sampling false negatives increases during training and be much larger value than that of random sampling,[1] false negatives give more significant impact.

2.4 Effect of Sub-Sampling

As discussed in Section 2.2, the sub-sampling technique in DNS gives a robustness against false negatives. To investigate the effectiveness of sub-sampling, we compare the accuracy of models with different value of $C = \{50, 150, 250, 400, 600, 800, 1,000\}$, and fixed other parameters. We report the results on Amazon-Books only in Figure 1 as we observe similar trends on other datasets.

Figure 1a shows Precision@10 with different size of candidates in ordinary sampling. In the figure, $C = 150$ and 250 gives the best, indicating that smaller or larger values of C give lower performance. We observe the followings:

- If there are few candidates, it is not possible to sample highly ranked items effectively. Thus, it is difficult to construct an accurate model.
- If there are many candidates, the probability of sampling a false negative is high. Thus, a model overfits to training data, making the model inaccurate.

On the other hand, as ideal sampling never samples false negatives, sub-sampling techniques just make it more difficult to choose highly ranked items in true negatives. Therefore, we can assume that the larger the candidate size, the better the performance. As expected, Figure 1b shows that the larger C, the better accuracy at Precision@10 in ideal sampling. Making a comparison between this result and that obtained by ordinary sampling (Figure 1a), we can argue that the sub-sampling size works as a parameter to balance the performance drop due to false negatives and the performance improvement due to rank-biased sampling.

[1] In our experiments, usual DNS is 15-30 times more likely to sample false negatives than random.

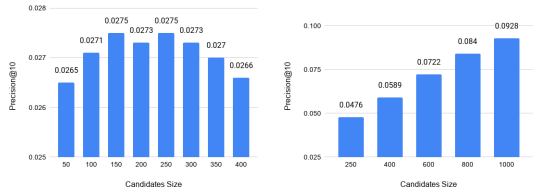
Table 1: Comparison of precision (P@10 and P@20) between ordinary and ideal sampler (OS and IS, respectively) in (a) random sampling and (b) dynamic sampling (DNS).

(a) Random sampling

Dataset	MovieLens10M		Amazon-Books		Yelp	
Metrics	P@10	P@20	P@10	P@20	P@10	P@20
OS	0.0525	0.0466	0.0222	0.0200	0.0204	0.0185
IS	**0.0531**	**0.4690**	**0.0243**	**0.0217**	**0.0208**	**0.0186**
Improvement	0.9%	1.0%	9.9%	8.5%	2.0%	0.5%

(b) Dynamic sampling (DNS)

Dataset	MovieLens10M		Amazon-Books		Yelp	
Metrics	P@10	P@20	P@10	P@20	P@10	P@20
OS	0.0552	0.0485	0.0275	0.0244	0.0273	0.0237
IS	**0.1473**	**0.0893**	**0.0924**	**0.0591**	**0.0952**	**0.0543**
Improvement	174.8%	84.1%	238.4%	144.0%	255.2%	133.0%



(a) Ordinary DNS **(b)** Ideal DNS

Figure 1: Precision@10 with different C on Amazon-Books.

3 UPPER BOUND OF MODEL ACCURACY

3.1 The Classification Error of Sampler

A recommender model aims at discriminating false negatives from other items and generating a list of them. On the other hand, ideal sampling can be taken as an algorithm in which the sampler keeps a complete list of false negatives for each user and does not sample them. These imply a simple fact that the list of false negatives kept by the sampler is exactly the same as the list of items that a recommender model should provide. That is why the goal of the recommender model is to discriminate false negatives from other data and try to include them in the list of recommendation as well as true positives. Therefore, we can argue that a model and a sampler actually share the same goal: identification of false negatives (*i.e.*, test data) from other unobserved data. A model tries to identify

them to recommend while a sampler tries to exclude them from sampled list.

Thus, a sampler can also be defined as a classification model D_s. The definition is almost the same as Equation (1). A sampler should sample items from unobserved data $I \setminus I_u$ but excludes the items which $D_s(i|u) = 1$ (*i.e.*, predicted as false negative) for each user. Therefore, the probability of sampling item $i \in I \setminus I_u$ for each user u is defined as follows:

$$p(i|u) = \begin{cases} 0 & D_s(i|u) = 1 \\ s(i|u) & otherwise, \end{cases}$$

$$s(i|u) \sim r'_\theta(i|u)^{-\gamma},$$

where $\gamma \geq 0$ is a polynomial degree. When $\gamma = 0$, this sampler is equal to random sampling, otherwise rank-biased sampling. Moreover, to compute efficiency and provide robustness against false negatives, we introduce sub-sampling technique in Section 2.4. We create a set of candidates C for each positive data with uniform distribution and then update the above $p(i|u)$ as defined in the following Equation (5):

$$p(i|u) = \begin{cases} 0 & D_s(i|u) = 1 \vee i \notin C \\ s(i|u) & otherwise, \end{cases} \tag{5}$$

$$s(i|u) \sim r''_\theta(i|u)^{-\gamma},$$

$$r''_\theta(i|u) := |\{j \in C : \hat{y}_\theta(j|u) \geq \hat{y}_\theta(i|u)\}|.$$

Recommendations using the list of false negatives of ideal sampler would achieve $e'_m \to e'_{m_{min}}$ (*i.e.*, it identifies test data from other unobserved data as much as possible) as the list of false negatives is that of positive samples in test data. Given this, the recommendation accuracy of the ideal sampler is clearly better than that of a model, which looks somehow strange. Thus, we come up with a simple theorem that we cannot define a sampling method that outperforms the classification performance of the model. When we define the overall classification error of sampler e'_s in the same way as Equation (3), the following theorem holds:

Theorem 1. *For the classification error of model for test data e'_θ and sampler e'_s , the following is always satisfied:*

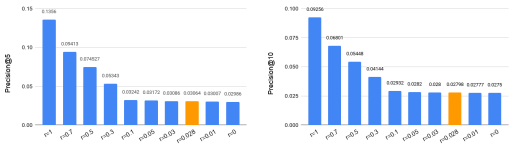
$$e'_\theta \leq e'_s$$

Proof 1. *If $e'_s \leq e'_m$, where e'_m is classification error of model, the model apparently can be copied from the sampler. Then $e'_m = e'_s$.*

Any sampling method can be extended to a model that predicts false negatives: random and rank-biased sampling can be extend to models that predict false negatives according to a uniform distribution, which is the same as random recommendation. In this case, the accuracy of the model exceeds that of these samplers.

3.2 Upper Bound Estimation

According to Theorem 1, the accuracy of the sampler cannot be better than that of the model. On the other hand, if the accuracy of the model is higher than that of the sampler, the sampler accuracy has room for improvement. If the accuracy of the sampler improves, it will be able to exclude more false negatives and expected to further improve the accuracy of the model. Based on this intuition, we hypothesize that a model accuracy is maximized when the accuracy of the model and that of a sampler match.



(a) Precision@5 **(b)** Precision@10

Figure 2: Sensitivity with regard to different r.

Hypothesis 1. *If e'_m is minimal, e'_m and the error of sampler e'_s meet the following condition:*

$$e'_m = e'_s.$$

Unfortunately, it is difficult to show this hypothesis mathematically as there is no closed-form solution or guaranteed convergence of solution in models we are interested in. Therefore, we experimentally show the above hypothesis.

In order to experimentally investigate the hypothesis, we employ a pseudo negative sampling method with $e'_s \neq 0$. We prepare an artificial sampling technique that would exclude some false negatives and randomly selected true negative items. This method can be thought of as ideal sampling contaminated with true negatives, which is closer to the practical sampling method. We refer to it as contaminated sampling. To compare a model accuracy with different e'_s, we use the ratio of false negatives among the items to be excluded $r \in [0, 1](= 1 - e'_s)$. If $r = 1$, it is equivalent to ideal sampling, and $r = 0$ is equivalent to ordinary sampling.

If a model performance (*i.e.*, $1 - e'_m$) increases monotonically as r increases, then we have experimentally shown that the hypothesis is valid as the maximum performance is clearly achieved when the error ratio of the model e'_m is equal to that of the sampler e'_s under conditions that do not violate Theorem 1.

Experimental Setup. We only consider the upper bound in DNS as ideal DNS shows significant improvement in Section 2.3. We set $r = \{1, 0.7, 0.5, 0.3, 0.1, 0.05, 0.03, 0.028, 0.01, 0\}$ and tune the size of candidates C for each r. We select the best result from $C \in \{50, 100, 150, 200, 250, 300, 350, 400, 500, 600, 800, 1,000\}$ for each r. Due to space limits, we only show the result on Amazon-Books as experimental results on other datasets are similar.

Results. Figures 2a and 2b show precision (@5 and @10) by varying r, respectively. We observe that the model accuracy e'_m gradually worsens as r is smaller, which supports the aforementioned hypothesis that a model accuracy is maximized when the accuracy of the model and the sampler exactly match. In addition, it suggests that the better the model accuracy, the more improvement can be expected from ordinary sampling as there are more room for improvement in a sampler accuracy.

4 GENERAL FRAMEWORK FOR NEGATIVE SAMPLING

In our framework, we formulate negative sampling method as a classification model and sample items based on Equation (5). Moreover, we introduce two techniques which make our method generalized and more robust against false negatives: **self-sampling** and **dynamic sub-sampling** shown in Algorithm 1.

Algorithm 1 Generalized Negative Sampling Algorithm

Input: training data S, a polynomial degree γ, the number of false negatives K, the min/max size of candidates C_{min}, C_{max}
Output: top-N recommendation list

```
Initialize model parameters θ
for epoch do
    Fetch mini-batch data D_pos from S
    for each data do
        Create candidates set C ⊂ I ~ Uniform
        Sample instances d_neg ~ p(i ∈ C|u),          ▷ Eq. (5)
    end for
    Calculate loss                                    ▷ Eq. (4)
    Update θ using Adam optimizer [11]
end for
```

4.1 Self Sampling

In Equation (5), we can choose any classification models for a sampler. For example, if we use linear models as they often outperform model-based recommender systems [3, 19], a model can achieve the same accuracy as them. Another idea is to train a sampler in the same way as a model, which can train a model and a sampler by alternating their roles. This learning method is interesting but it requires additional computational complexity to train a sampler.

In our method, we use the recommender model itself as classification model in a sampler (*i.e.*, $D_s(i|u) = D_m(i|u)$) as it always meet the condition of Hypothesis 1 and does not need to train the sampler. Another advantage of self-sampling is that it can be applied to any model-based recommendation systems. While we only consider Biased MF in our experiments, the framework can be generally applied to other models (*e.g.*, FM [15], CDAE [21], and NeuMF [7]) which use negative sampling for training.

Dynamic Self Sampling. To define a classification model of sampler $D_s(i|u)$, we need to know $|I'_u|$ in Equation (1), but the number of false negatives is not clear in practical situation. Thus, we give the number of false negatives K artificially instead of $|I'_u|$. Thus, a sampler considers items in top-K ranking as false negatives and samples higher-ranking items by excluding them.

However, it is not reasonable to fix K to a large value at the early stages of training. In our framework, when we exclude some false negatives, we also exclude many true negatives with large error as it removes the highest-ranking items. In other words, if we use large K while the accuracy is low, the learning process does not work well. Thus, we employ a strategy which gradually increases the number of items that a sampler excludes from sampling. At epoch e in training, the number of items to exclude k_e is set to:

$$k_e = \lfloor \frac{e}{E} K \rfloor,$$

where E is the number of training epoch. In our experiments, we further trained a model as $k_s = 0$ in the initial constant epoch (*e.g.*, 10). This could be pre-training by DNS.

4.2 Dynamic Sub-sampling

In Section 2.4, we discussed that when the probability of sampling false negatives is small, increasing the size of sub-sampling $|C|$ improves the accuracy. In our framework, we improve the accuracy by increasing C as well as excluding some false negatives. In order to keep consistency with the aforementioned dynamic self sampling strategy, we gradually increase $|C|$ as k_e increases. Thus, let the number of candidates at epoch e be C_e, which is set to:

$$C_e = C_{min} + \frac{C_{max} - C_{min}}{E},$$

where C_{min} and C_{max} are the initial and the maximum value of C, respectively.

Table 2: Some statistics on datasets used in our experiments.

Dataset	#Users	#items	#Train	#Val & #Test
MovieLens10M	48,405	10,042	8,293,623	484,050
Amazon-Books	18,235	61,302	1,548,723	182,350
Yelp	15,030	51,078	1,055,334	150,300

5 EXPERIMENTS

5.1 Experimental Settings

We employ the following three publicly available datasets: **MovieLens10M**,[2] **Amazon-Books**,[3] and **Yelp**[4] (see Table 2 for some statistics on these datasets). Although all of them are originally 5-star rating data, we convert them into binary implicit feedback data. Additionally, in Amazon-Books, the data of all products is so large that we selected only the review data (from 2016) of the "book", which is the largest one among all the categories. We employ leave-K-out protocol to split them into training, validation, and test data. For each user, the K most recent true positive interactions (*i.e.*, three or more ratings) are used as test data. Then, the subsequent K most recent interactions including false positives (*i.e.*, below two ratings) are used as validation data, and the rest are training data.

Evaluation Measure: We evaluate our proposed approach with Precision@N, Recall@N, and nDCG@N [10] (N = 5 and 10).

Recommender Model: We employ Biased MF [13] as a recommender model. In our experiment, we set b_u, b_i, and μ to 0 for simplicity. We fix the embedding size and learning epoch to 128 and 300, respectively, which is reasonable and large enough. Under these constraints, we optimize parameters for our experiments.

Baselines: We compare our proposed approach with the following sampling methods:

- **Weight** [8]: A non-sampling matrix factorization method which applies uniform weight w on all unobserved data. Recent work on non-sampling strategy often uses the same strategy [2] and provides better recommendation.
- **Random**: Our experiments uses the same technique proposed in BPR [17]. We set the number of negative items to be sampled for each positive item to 1.
- **DNS** [22]: A mixture technique of random and rank-biased sampling. Note that our preliminaly experiments show that DNS outperforms the state-of-the-arts, KGPolicy [20] and SRNS [5], motivating us to employ DNS as one of baselines.

Parameter Settings: We identify optimized parameters via random search.

[2] https://grouplens.org/datasets/MovieLens/10m/
[3] https://nijianmo.github.io/amazon/index.html
[4] https://www.yelp.com/dataset

Table 3: Comparison of performance on (a) MovieLens10M, (b) Amazon-Books, and (c) Yelp.

(a) MovieLens10M

Metrics	Precision		Recall		nDCG	
	@5	@10	@5	@10	@5	@10
Weight	0.0536	0.0490	0.0268	0.0490	0.0558	0.0516
Random	0.0585	0.0525	0.0293	0.0525	0.0610	0.0560
DNS	0.0619	0.0552	0.0310	0.0552	0.0650	0.0592
Proposed	**0.0625***	**0.0556***	**0.0313***	**0.0556***	**0.0657***	**0.0598***
Improvement	1.0%	0.7%	0.9%	0.7%	1.1%	1.0%

(b) Amazon-Books

Metrics	Precision		Recall		nDCG	
	@5	@10	@5	@10	@5	@10
Weight	0.0270	0.0249	0.0135	0.0249	0.0275	0.0258
Random	0.0240	0.0221	0.0120	0.0221	0.0250	0.0235
DNS	0.0298	0.0275	0.0152	0.0275	0.0307	0.0286
Proposed	**0.0306***	**0.0278***	**0.0154***	**0.0278***	**0.0313***	**0.0290***
Improvement	2.6%	1.0%	2.6%	1.0%	2.6%	1.3%

(c) Yelp

Metrics	Precision		Recall		nDCG	
	@5	@10	@5	@10	@5	@10
Weight	0.0300	0.0264	0.0150	0.0264	0.0315	0.0281
Random	0.0224	0.0222	0.0109	0.0222	0.0231	0.0217
DNS	0.0308	0.0273	0.0154	0.0273	0.0328	0.0295
Proposed	**0.0313***	**0.0275***	**0.0156***	**0.0275***	**0.0333***	**0.0298***
Improvement	1.6%	0.7%	1.2%	0.7%	1.5%	1.0%

5.2 Experimental Results

Tables 3a, 3b, and 3c show the results on MovieLens10M, Amazon-Books, and Yelp, respectively. We observe that our proposed method consistently improves the performance with statistical significance (for $p < 0.05$, marked with "*" in these tables) compared with the best baseline DNS (underlined scores). As discussed in Section 3.2, we do not observe significant difference between the upper bound of the accuracy and that obtained by DNS.. For example, in Amazon-Books, about 3% and 2% differences are estimated for Precision@5 and Precision@10, respectively. Table 3b shows that our method successfully improves Precision@5 and Precision@10 by 2.6% and 1.0%, respectively. Thus, our method can approach the upper bound more closely than DNS.

For baselines, we observe that DNS consistently outperforms the others. Weighted MF does not work well in MovieLens10M but outperforms random sampling in Amazon-Books and Yelp. In the case where the number of items is large, random sampling is not effective as it is very difficult to sample data that improves training. It is also notable that the improvement in @5 is more significant than @10. The results are consistent with the analysis on the potential improvement with higher top-N (see Section 3.2).

Limitations. We observe that the number of false negatives affects a model accuracy. Although we set the same number of false negatives for all users in our experiments, it can be actually different user by user. In order to address this, it is necessary to estimate the optimal number of false negatives for each user.

6 CONCLUSION

We first conducted preliminary experiments to analyze the impact of false negatives in matrix factorization. By introducing the concept of ideal sampling, we could verify the classification ability of the model without false negatives and identified that they could be a specific problem with rank-biased negative sampling.

Moreover, based on a simple fact that both recommender models and negative sampling can be regarded as models for classifying unobserved data into false and true negatives, we argued that the classification ability of negative sampler cannot outperform that of recommender. Thus, we could estimate the actual upper bound of improvements by avoiding false negatives. To approach the upper bound, we developed a general framework for negative sampling by introducing two robustness against false negatives: self-sampling and dynamic sub-sampling. In future work, we plan to develop a more effective sampler with better classification accuracy.

Acknowledgment. We thank Tokyo ACM SIGIR Chapter committee members for their helpful and insightful comments,

REFERENCES

[1] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A Generic Collaborative Filtering Framework based on Generative Adversarial Networks. In *CIKM*. 137–146.
[2] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient Neural Matrix Factorization without Sampling for Recommendation. *ACM TOIS* 38, 2 (2020), 14:1–14:28.
[3] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. 2021. A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research. *ACM TOIS* 39, 2 (2021), 20:1–20:49.
[4] Jingtao Ding, Yuhan Quan, Xiangnan He, Yong Li, and Depeng Jin. 2019. Reinforced Negative Sampling for Recommendation with Exposure Data. In *IJCAI*. 2230–2236.
[5] Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. 2020. Simplify and Robustify Negative Sampling for Implicit Collaborative Filtering. In *NeurIPS*.
[6] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommendation. In *SIGIR*. 355–364.
[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[8] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *ICDM*. 263–272.
[9] Dietmar Jannach, Lukas Lerche, and Markus Zanker. 2018. Recommending based on Implicit Feedback. In *Social Information Access*. Springer, 510–569.
[10] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM TOIS* 20, 4 (2002), 422–446.
[11] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[12] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *KDD*. 426–434.
[13] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42 (2009), 30–37.
[14] Dae Hoon Park and Yi Chang. 2019. Adversarial Sampling and Training for Semi-supervised Information Retrieval. In *WWW*. 1443–1453.
[15] Steffen Rendle. 2010. Factorization Machines. In *ICDM*. 995–1000.
[16] Steffen Rendle and Christoph Freudenthaler. 2014. Improving Pairwise Learning for Item Recommendation from Implicit Feedback. In *WSDM*. 273–282.
[17] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[18] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *RecSys*. 240–248.
[19] Harald Steck. 2019. Embarrassingly Shallow Autoencoders for Sparse Data. In *WWW*. 3251–3257.
[20] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *WWW*. 99–109.
[21] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*. 153–162.
[22] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing Top-N Collaborative Filtering via Dynamic Negative Item Sampling. In *SIGIR*. 785–788.