NEGATIVE SAMPLING FROM THE GROUND UP: A REDESIGN FOR GRAPH-BASED RECOMMENDATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Negative sampling is an important yet challenging component in self-supervised graph representation learning, particularly for recommendation systems where user-item interactions are modeled as bipartite graphs. Existing methods often rely on heuristics or human-specified principles to design negative sampling distributions. This potentially overlooks the usage of an underlying "true" negative distribution, which we might be able to access as an oracle despite not knowing its exact form. In this work, we shift the focus from manually designing negative sampling distributions to a more principled method that approximates and leverages the underlying true distribution from the ground up. We expand this idea in the analysis of two scenarios: (1) when the observed graph is an unbiased sample from the true distribution, and (2) when the observed graph is biased with partially observable positive edges. The analysis result is the derivation of a sampling strategy as the numerical approximation of a well-established learning objective. Our theoretical findings are also empirically validated, and our new sampling methods achieve state-of-the-art performance on real-world datasets. Our code can be downloaded at `https://anonymous.4open.science/r/NS-Graph-Rec/`.

1 INTRODUCTION

Self-supervised graph representation learning has wide applications in modern recommendation systems. To obtain high-quality embeddings for users and items, the typical method is to model past user-item interactions as a bipartite graph. Node embeddings are often learned in such a way that the distance between a user's embedding and an interacted item's embedding indicates the likelihood of an edge between the two objects.

Negative sampling is an important step in this learning paradigm: as shown in Figure 1(a), for a given anchor node u, a node v^- needs to be sampled from a certain distribution q^-, so that node pair (u, v^-) can be treated as a "negative edge" in training, together with the observed positive edge (u, v^+), for training the model via binary loss. Negative sampling is important since it generates all negative samples in the training dataset by certain human-specified rules, which can have huge influence on learned representations. This also applies to more general graphs outside of the bipartite user-item graphs in recommendations.

Negative sampling is not only important but also challenging. Existing works have proposed many heuristics, including random negative sampling (RNS) (Rendle et al., 2012), popularity-based negative sampling (PNS) (Mikolov et al., 2013), hard negative sampling (HNS) (Ying et al., 2018), GAN-based negative sampling (GAN-based NS) (Chae et al., 2018), and in-batch negative sampling (in-batch NS) (Wu et al., 2021), *etc*. More recent theoretical studies (Yan et al., 2024; Yang et al., 2020b) have proposed ideal properties of node embeddings that a good negative distribution should encourage, and then design negative distributions accordingly.

These previous works leave open a critical question: should we consider negative sampling to be a procedure that is primarily up to human heuristics or design, or can we, instead, build the procedure from the ground up — by treating it as a rigorous numerical approximation of certain well-established optimization objective? In this paper, we argue to support the latter, and show a path to achieve it.

We start by formulating and highlighting the concept of the true distribution of negative samples p^-, whose objective existence is independent of human choices. As a motivating example, consider that we are in an ideal world where we have enough resources and capability to survey the true opinion of each user u's potent to like each item v in the entire content pool. This true opinion

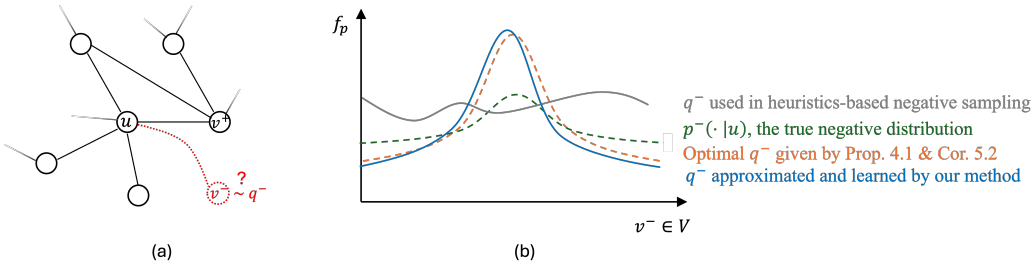

Figure 1: **(a)** The negative sampling problem aims to find an optimal distribution q^- to sample negative node v^- from. **(b)** Illustrating the key distributions and idea in our analysis: The green curve $p^-(|u)$ is the real negative distribution for v^- *w.r.t.* anchor node u, which we argue to exist objectively. Our Prop. 4.1 shows that the optimal q^- should be $p^-(|u)$ reweighed by an exponential factor that relies on the model's output in current epoch. We observe that q^- proposed in previous works, denoted as the gray curve, is mostly heuristics-based and does not leverage the optimal q^-, and thus correlates little on distribution landscape. In comparison, our negative sampling seeks to learn and approximate q^-, denoted as the blue curve, albeit with inevitable deviation. The bell-shaped curves are only for illustrative purpose and do *not* reflect any assumption about the distributions.

value can be technically formulated as $P(y = 1|e = (u, v))$, where y is the true class label of the user's like. The Bayes rule gives us distribution $p^+ = P(e|y = 1) \propto P(y = 1|e)P(e)$ and distribution $p^- = P(e|y = 0) \propto (1 - P(y = 1|e))P(e)$, where $P(e)$ is the node pair prior. In the above formulation, p^- is the true negative distribution that we want to emphasize: every graph from a well-defined domain in practice should have such a p^- that objectively exists. We will provide more rigorous definitions in Section 3.

Of course, the exact form of p^- is unknown in most problems, otherwise the classification problem is already algebraically solvable by reversely deriving $P(y = 1|e)$ from p^-, in which case there is no need to do sampling and training. However, p^- being unknown does not necessarily preclude us from being able to access (samples from) p^- as an oracle (though again in many cases there is distribution shift in observation, which we will have a dedicated section to address). The reason that we may still need to generate negative samples from a different distribution than p^- is primarily to reduce variance of learning, *i.e.* to speed up convergence with fewer samples. Some existing analysis (Mukherjee et al., 2011) of classical boosting algorithms such as AdaBoost helps further illustrate this point.

The true negative distribution p^- guides our principled derivation of a good *proposal* negative distribution q^-. Illustrated in Figure 1(b), we argue that a good q should be grounded and focused on leveraging the noted true (albeit unknown) negative distribution p^- — including understanding how p^- factors into the expression of a well-established optimization goal, as well as how it can be approximated or recovered if distorted. In fact, an interesting observation is that most existing negative sampling methods made this assumption implicitly without fully exploiting its usage. See Appendix A for more discussion.

Our work. In this work, we analyze how to build towards an optimal sampling strategy, by shifting the focus from proposing distributions into crafting sampling procedures based on the argument and the leverage that a true p^- exists, which further expands into two cases. In the first case, we assume that the observed graph is an unbiased sample from the true distribution. q^- can then be derived by drawing its connection to a well-established objective that involves p^-. It turns out that the resulted sampling strategy can be interpreted as a form of adaptive hard negative sampling.

In the second case, we assume that the observed graph is a biased sample from the true distribution, and that the positive edges may be only *partially* observable. This case is underexplored in graph learning's literature but has wide applications. For example, in recommendations a positive edge represents a user's manifested interest in an item logged by the platform; however, the absence of an edge can either mean the user truly dislikes the item, or that the user just has not been offered the chance to interact with the item but would have liked it if so. In fact, due to the sparse nature of many complex systems only a small fraction of positive edges may be observable.

To address the second case, the idea is to first derive an unbiased empirical risk estimator depending on *true* distributions, and then convert it into a form that depends only on *observed* distributions, up to some calibration. The calibration, which was deemed a bottleneck in more general settings, can be facilitated by graph topology in a learnable manner. We further validated this new sampling method on real-world datasets and achieved state-of-the-art results.

Scope. We primarily consider negative sampling in the context of recommendation and link prediction tasks, following most existing literature on this topic. See Appendix B for more discussion.

Contributions. We make the following contributions in this work:

- We propose to fundamentally shift the mode of thinking from proposing negative distributions to principally approximating and utilizing the true distributions in sampling methods for graph representation learning in both recommendations and beyond.
- We theoretically derive and analyze the forms that the optimal sampling strategy should follow, under the assumption that the positive edges are sampled unbiasedly from the true distribution, or that the positive edges are only partially observable.
- We further validate our theorems and new sampling method on real-world data and observe their significant improvements over previous methods.

2 Related Work

Self-supervised Learning on Graphs. Self-supervised Learning on Graphs aims to train a graph encoder using both the original graph and often some stochastically augmented data, with many applications including recommendations (Shuai et al., 2022), drug discovery (Wang et al., 2021), anomaly detection (Wang et al., 2024), *etc.* Fancy data augmentations are less common for industry-level recommendations due to scalability and efficiency concerns (Yu et al., 2022). In practice, positive samples are usually directly sampled from the edge set or via random walks (Ying et al., 2018), and negative samples by fixing an anchor node and then drawing the other from a certain negative distribution (Yang et al., 2020a).

Negative Sampling. Negative sampling was originally proposed as a component in noise contrastive estimation (Gutmann & Hyvärinen, 2010; 2012), and was soon applied to language modeling (Mikolov et al., 2013; Mnih & Teh, 2012) for approximating softmax. It emerged in graph's context when node embedding methods (Perozzi et al., 2014) took inspirations from language modeling. Since then, it has been widely applied in self-supervised graph learning, and in recommendations (Yang et al., 2020a), and soon became a standalone research topic due to its importance.

Existing works on negative sampling have proposed many heuristics, including random negative sampling (RNS) (Rendle et al., 2012; Bordes et al., 2013), popularity-based negative sampling (PNS) (Mikolov et al., 2013; Perozzi et al., 2014), hard negative sampling (HNS) (Huang et al., 2021; Lai et al., 2024) , GAN-based negative sampling (GAN-based NS) (Chae et al., 2018; Wang et al., 2018), and in-batch negative sampling (in-batch NS) (Wu et al., 2021; Chen et al., 2020; Zhao et al., 2021), *etc*. More recent theoretical works (Yan et al., 2024; Yang et al., 2020b; Robinson et al., 2021) proposed properties that a good negative distribution should satisfy, such as monotonicity and accuracy. Shi et al. (2023) analyzes the relationship between hard negatives and BPR loss. (Petrov & Macdonald, 2023) studies over-confidence in negative sampling, and (Chen et al., 2023a) studies the comparison between negative sampling and non-sampling.

Some graph embedding models propose their own negative sampling procedures, *e.g.*, GraphSAGE (Hamilton et al., 2017) and DeepWalk Perozzi et al. (2014) both use PNS; PinSAGE (Ying et al., 2018) uses a mixture of in-batch NS and HNS customized for large-scale training. In comparison, some other GNN models did not specify their accompanied negative sampling, such as GCN (Kipf & Welling, 2016), GAT Veličković et al. (2017), MPNN Gilmer et al. (2017), SGC (Wu et al., 2019). (Ma et al., 2024) provides a comprehensive survey on negative sampling in recommendations.

3 Problem Formulation and Definitions

3.1 Problem Formulation

The standard task of self-supervised learning on a graph is defined as the following. We are given a graph $G = (V, E)$ where V is the node set, E is the edge set. Denote by $x_u \in \mathbb{R}^m$ the raw features of node $u \in V$. The goal is to learn a graph encoder f defined as a mapping $\mathbb{R}^m \to \mathbb{R}^n$, so that $f(x_u)$ can best represent node u and be used in downstream tasks. f is usually parameterized by a neural network with parameters θ, learned by minimizing the following objective:

$$J = \sum_{\substack{(u,v)\in E,\\ v_1^-,\ldots,v_k^- \sim q^-}} \left[L^+(u,v) + \frac{\tau}{k} \sum_{i=1}^{k} L^-(u, v_i^-) \right] \tag{1}$$

$$\text{where} \quad L^+(u,v) = g^+(f(x_u), f(x_v)), \quad L^-(u,v^-) = g^-(f(x_u), f(x_{v^-})) \tag{2}$$

which traverses every edge (u, v) in E, picking k negative nodes from the proposal negative distribution q^- for each edge, and collecting the corresponding positive and negative losses. g^+ and g^- are simple functions that measure dissimilarity and similarities between the f-encoded node embeddings $f(x_u)$ and $f(x_v)$, respectively. τ is a weighing constant for analysis purpose, assumed to be 1 by default.

The negative sampling problem asks (as consistent with Yang et al. (2020b); Yan et al. (2024)):

($*$) ***What's the best form of q^- to use in the above objective J?***

3.2 Probabilistic Setup

First, all probabilities are defined in the sample space $\{(e, y, s)\}$ which has the following meaning: (1) Random variable $e = (u, v) \in V \times V$ is a node pair in the graph. (2) Random variable $y \in \{0, 1\}$ is the class indicator variable for ground truth. $y = 1$ means that the associated node pair e is a truly positive edge, and $y = 0$ otherwise. (3) Random variable $s \in \{0, 1\}$ is the observation indicator variable. $s = 1$ means the edge is observed, and $s = 0$ otherwise.

We can define several distributions based on the specified sample space. $p^+ = P(e|y = 1)$ and $p^- = P(e|y = 0)$ are the true positive distribution and true negative distribution respectively. $\hat{p}^+ = P(e|s = 1)$ and $\hat{p}^- = P(e|s = 0)$ are the observable positive distribution and the observable negative distribution respectively. $\pi^+ = P(y = 1)$ and $\pi^- = P(y = 0)$ are the class priors for the ground truth. $p = P(e)$ is the node pair prior, assumed to be uniform by default, *i.e.* all node pairs are assigned equal weight of consideration initially. $P(y = 1|e)$ represents the link predictor or recommender that we wish to obtain ultimately.

If the data is unbiasedly sampled then $\hat{p}^+ = p^+$, $\hat{p}^- = p^-$. We hold this assumption until Section 6 when we address the case of biased observation. Also notice that G can either be viewed as containing a set of edges E drawn from distribution $\hat{p}^+$, or equivalently as containing a set of negative edges, $V \times V \backslash E$, drawn from distribution $\hat{p}^-$.

As a convention of notation: since $p^+ = p^+(e) = p^+(u, v) = P(e|y = 1) = P(u, v|y = 1)$, p^+ is essentially a joint distribution of nodes u, v, *i.e.* $p^+ : V \times V \to [0, 1]$. Hence we can define its corresponding marginal distribution $p^+(u)$, and conditional distribution $p^+(\cdot|u)$. We further use $(u, v) \sim p^+$ to denote that u, v are drawn from the joint distribution, and use $u \sim p^+$ to denote that u is drawn from the marginal distribution. Similar conventions apply to p^-, $\hat{p}^+$, and $\hat{p}^-$.

The following proposition gives the generalized objective that negative sampling problem optimizes.

Proposition 3.1. *J is the empirical estimation of the following expected risk term*

$$R_1(f) = \mathbb{E}_{u\sim p^+}[\mathbb{E}_{v\sim p^+(\cdot|u)}[L^+(u, v)] + \mathbb{E}_{v^-\sim q^-}[L^-(u, v^-)]] \tag{3}$$

3.3 Difference between q^-, $p^-(\cdot|u)$, and $\hat{p}^-(\cdot|u)$

It is crucial to distinguish the three negative distributions noted above: $q^-, p^-(\cdot|u), \hat{p}^-(\cdot|u)$, which are all defined over node set V. q^- is the proposal negative distribution whose optimal form we seek to find, which may or may not depend on the anchor node u though. That is why in the notation we suppress the dependence of q^- on any other entities for the time being. $p^-(\cdot|u)$ is the true negative distribution given anchor node u fixed, which we cannot directly observe if the observations are biased. $\hat{p}^-(\cdot|u)$ is the observed negative distribution, the distribution that our data (*i.e.* the observed negative edges) are directly sampled from.

Our ultimate goal in this paper is to construct q^- from $\hat{p}^-(\cdot|u)$, as introduced in Section 1. Also notice that although the best form of q^- depends on p, it does not have to be exactly the same as p.

4 Unbiased Observations

Roadmap. In this section, we show how the best form of q^- can be principally derived in two steps. First, we introduce another objective $R_2(f)$, in addition to the $R_1(f)$ defined above. By explaining $R_2(f)$, we will show why it is a well-established objective to optimize for training. Second, once we've established the importance of $R_2(f)$, we can show there is only one specific q^- that makes $R_1(f)$ equivalent to $R_2(f)$, thereby proving the optimality of our design for q^-.

What is $R_2(f)$?
Given a graph $G = (V, E)$ with distributions p^+ and p^- as defined in Section 3, consider the following risk term $R_2(f)$ that we wish to minimize, associated with encoder f that has parameters θ:

$$R_2(f) = \mathbb{E}_{(u,v)\sim p^+}[-\log\ p_f(v|u)] \tag{4}$$

$$p_f(v|u) = \frac{e^{g(f(x_u),f(x_v))}}{e^{g(f(x_u),f(x_v))} + \tau\mathbb{E}_{v^-\sim p^-(\cdot|u)}e^{g(f(x_u),f(x_{v^-}))}} \tag{5}$$

where $p_f(v|u)$ is the generalized softmax probability of node v given anchor node u, based on f's output. g can be any similarity measurement such *e.g.* dot product, and at this moment it is to be distinguished from the g^+ and g^- in problem formulation. See more discussion in Appendix C.2 for ensuring p_f to be a proper distribution.

The definition of $p_f(v|u)$ above has shown up as a popular objective in many previous works Robinson et al. (2021); Rusak et al. (2024); Oord et al. (2018); Tian et al. (2020); Chen et al. (2020). For example, if we set $\tau = 1$, then it is the expectation of the classic noise-contrastive estimation (NCE) loss in Robinson et al. (2021)'s Eq.(1); if we set $\tau = |V| - 1$, it becomes the expectation of the InfoNCE loss - see Rusak et al. (2024)'s Eq.(1) or the original paper Oord et al. (2018)'s Eq.(4). .

Also, crucially note that the negative distribution in the denominator of Eq.5 is p^-, instead of q^-. This is because we desire the probability of each edge to be as large as possible — calibrated against the probabilities of *all* other possible negative edges sampled from the *true* negative distribution p^-, rather than against edges from an arbitrary proposal distribution q^-. Using p^- for calibration reflects the competition that positive edges face from the entire set (distribution) of negative edges, which represents the general objective we seek to optimize.

Why is $R_2(f)$ an important objective to optimize?
$R_2(f)$ can be further interpreted by writing down its empirical estimator:

$$\hat{R}_2(f) = \frac{1}{n}\sum_{\{(u_i,v_i)\}_{i=1}^n\sim\hat{p}^+}[-\log\ p_f(v|u)] = \frac{1}{|E|}\sum_{(u,v)\in E}[-\log\ p_f(v|u)] \tag{6}$$

which has been widely used in contrastive learning literature as explained above. Besides, minimizing $\hat{R}_2(f)$ is equivalent to maximizing the probability of the following random graph model, assuming uniform node prior $P(u)$. To see this, note that

$$P(G) = \prod_{u,v\in V} P(u,v)^{p^+(u,v)} = \prod_{u,v\in V}[p_f(v|u)P(u)]^{p^+(u,v)} = \mathrm{Z}\prod_{u,v\in V}[p_f(v|u)]^{\mathbb{1}_{\{(u,v)\in E\}}} \tag{7}$$

where $\mathrm{Z} = \prod_{u,v\in V} P(u)^{p^+(u,v)}$ is a constant. Taking log of Eq. 7 gives to $-\hat{R}_2(f)$.

What q^- can make $R_1(f)$ equivalent to $R_2(f)$?
If our ultimate goal is to minimize $R_2(f)$, what can we tell about the sampling problem defined in Section 3? The following proposition shows that there exists a unique choice of q^- which ensures that optimizing the $R_1(f)$ in Eq. 4 is identical to optimizing the $R_2(f)$ in Eq. 3.

Proposition 4.1. $\nabla_\theta R_1(f) \equiv \nabla_\theta R_2(f)$ *if and only if the following conditions hold:*

- *For each $u \in V$ fixed,* $q^-(v) \propto p_f(v|u)p^-(v|u)$
- $g^+ = -g, g^- = g$

Since $\nabla_\theta R_1(f) \equiv \nabla_\theta R_2(f) \Rightarrow \arg\min_\theta R_1(f) \equiv \arg\min_\theta R_2(f)$, this proposition shows that to optimize $R_2(f)$ the best proposal negative distribution q^- to be used in Eq. 1 should take the simple form of $q^- \propto p_f(\cdot|u)p^-(\cdot|u)$. To approximate $R_1(f)$, g^- should be further reweighed by the partition constant $z_u = \mathbb{E}_{v^-\sim p^-(\cdot|u)}[p_f(v^-|u)]$, which however does *not* need to be explicitly estimated as it cancels off with the z_u for normalizing q^- in sampling. See Appendix C.3 for further discussion.

To analyze $p_f(v|u)$'s implications, first notice that the equivalence above is defined between the two gradients, meaning that the property holds throughout the training of f. In other words, $p_f(v|u)$ is a changing term during the training, which is calculated based on the output of the model f in the current training epoch. Also notice that a large $p_f(v|u)$ means that the current model f leans towards classifying (u, v) as positive. Therefore, using $p_f(v|u)$ to reweigh the observed (true) negative distribution p^- is a precise form of *adaptive hard* negative sampling. This theoretical result also

intriguingly relates in spirit to some of the previous heuristic-driven sampling methods for adaptive hard negative sampling (Zhang et al., 2013; Robinson et al., 2021; Lai et al., 2024).

Implementation According to Proposition 4.1, we need to compute two terms, p^- and p_f, in order to draw samples from the proposal distribution q^-.To efficiently estimate p^-, we resort to the relationship $p = p^+\pi^+ + p^-\pi^- \Leftrightarrow p^- = (p - p^+\pi^+)/\pi^- = (p - \hat{p}^+\pi^+)/\pi^-$. This allows us to exploit the sparsity of observed $\hat{p}$ in practice. Refer to Section 3.2 for more definitions. Here, the positive prior π^+ is a hyperparameter to tune, and $\pi^- = 1 - \pi^+$. For approximating $\hat{p}^+$, it is typical to use the Monte Carlo method (Robert, 1999), which averages the delta function over all observed neighboring edges: $\hat{P}^+(\cdot|u) \simeq \frac{1}{|\mathcal{N}(u)|}\sum_{v\in\mathcal{N}(u)}\delta_{u,v}(\cdot)$, where $\delta_{u,v}$ is the Dirac delta function, and $\mathcal{N}(u)$ is the neighbors of u. To estimate p_f, we replace the expectation in its definition of Eq. 5 by sampled average.

5 Biased Observations

This section presents a sampling method for the bias case where only *some* positive edges are observable. Section 5.1 motivates the bias setup. Section 5.2 introduces an empirical estimator and its corresponding sampling method. Section 5.3 further describes a graph-learning-based method for estimating a key term in our sampling method.

5.1 Fomulating the Bias

Many of the real-world graphs have this interesting property of observation bias: while the observation of an edge often signifies a reliably logged interaction, the absence of an edge could either mean that the edge does not exist in reality, or that the existence of the edge just has not been testified.

For example, in recommendations, the absence of a user-item edge can either mean that the user truly dislikes the item, or that the user just has not been offered the chance to interact the item – but would have liked it if so. This also applies to other types of interactions such as social interactions and protein interactions, and patient-disease diagnosis relations.

To formulate this bias, recall from Section 3 definitions of the random variables: we consider probabilities in the sample space $\{(e, y, s)\}$ where $e \in V \times V$ is a random node pair, $y \in \{0, 1\}$ is the true class variable, and $s \in \{0, 1\}$ is the observation indicator variable.

It is important to distinguish the meaning between y and s. As an example in video recommendations, $e = (u, v)$ represents a user u and a video v that we consider. $y = 1$ means that u *would have* liked v if u is offered the chance to view v. Also refer to the "ideal world" explanation in Section 1. $s = 1$ means the fact that u likes v is actually logged by the platform, usually by judging from some predefined interaction metrics by the platform such as the actual action of clicking on the "like" button, long video viewing time, positive comment below the video, *etc.*

The observation bias that we consider stipulates that $s = 1$ only when $y = 1$, or equivalently, $P(s = 0|y = 0) = 1$ by contrapositive. We further define $\phi_{u,v} = P(s = 1|y = 1, e = (u, v))$, which is the probability that edge e gets observed conditioned on itself being a true positive. $\phi_{u,v}$ is also called propensity score in some literature.

5.2 Unbiased Estimator From Biased Observations

In this case, directly applying the sampling method over the observable distributions is problematic, because both the p^+ in the problem definition (Eq.3), and the p^- in $q^- \propto p_f(\cdot|u)p^-(\cdot|u)$, are now unobservable. If we directly replace them by $\hat{p}^+$ and $\hat{p}^-$, further corrections are needed. In fact, we will show that in this case not only $\hat{p}^-$ needs to be corrected, but so does $\hat{p}^+$. In other words, in order to approximate $R_2(f)$ under biased observation, we need to build two proposal distributions on top of $\hat{p}^+$ and $\hat{p}^-$: a proposal *positive* distribution q^+, and a proposal negative distribution q^-. The following theorem elaborates this idea.

Theorem 5.1. *Assume $L^+ = -L^-$, then*

$$R_2(f) \equiv \mathbb{E}_{u\sim\hat{p}^+}[\mathbb{E}_{u\sim q^+}[\beta_1 L^+] + \mathbb{E}_{u\sim q^-}[\beta_2 L^-]] \tag{8}$$

where

- $c = P(s = 1|y = 1)$, $\phi_{u,v} = P(s = 1|y = 1, e = (u, v))$;
- $q^+(v) \propto [\phi_{u,v} p_f(v|u) + \beta_1]^{-1}\, \hat{p}^+(v|u)$
- $q^-(v) \propto [\phi_{u,v}\, p_f(v|u)]^{-1}\, \hat{p}^-(v|u)$

- $\beta_1 = \frac{\pi^+ c(1-c)}{\pi^-}$, $\beta_2 = \frac{(1-\pi^+ c)c}{\pi^-}$

(Note that $\hat{p}^+, \hat{p}^-, \pi^+, \pi^-$ have been defined in Section 3.1; $p_f(v)$ has been defined in Eq. 15.)

Theorem 5.1 directly gives the desired form for the new sampling method that best approximates $R_2(f)$. We would provide pseudocode for the full algorithm at the end of this section. Similar to the unbiased case, here we also see p_f shows up as a reweight to the observed distributions, which carries rich implications – see previous discussion under Proposition 4.1.

Implementation. To implement this biased case, we first notice that both c and π^+ do not depend on v, and $\pi^+ c = P(s = 1)$ is a value that can be directly estimated from the data by $\frac{2|E|}{N(N-1)}$. π^+ remains to be a sampling hyperparameter to tune, and $\pi^- = 1 - \pi^+$. To efficiently approximate $\hat{p}^-$, we resort to the relationship $p = p^+\pi^+ c + p^-(1-\pi^+ c) \Leftrightarrow \hat{p}^- = (p - \hat{p}^+\pi^+ c)/(1-\pi^+ c)$, which allows us to exploit the sparsity of $\hat{p}$ in practice. Refer to Section 3.2 for more definitions. The sampling complexity remains the same as that in Sec. 4 — also same as uniform negative sampling.

The only unknown relying on v is the propensity score function $\phi_{u,v}$, discussed in next subsection.

5.3 Learning-based Estimation of Propensity Scores Using Graph Features

Here we introduce a learning-based method for approximating the propensity function $\phi_{u,v}$. We will first discuss its interpretation, then explain how it can be approximated by a graph encoder parameterized by a neural network.

Interpretation. $\phi_{u,v} = P(s = 1 | y = 1, e = (u, v))$ is essentially the observation bias of edge (u, v), also known as *exposure bias* in some literature. Here, our key observation is that the existing literature discussing these biases essentially establishes their strong correlation with various *graph-based features*, such as (1) Popularity bias (Zhang et al., 2021), which may be quantified by node degree d_u, d_v (2) Position bias Chen et al. (2023b), which may be quantified by the output of distance function g on the node pair (u, v). In addition, we also conjecture that the general structural information of the u, v may also be correlated factors.

Although these bias terms can be easy to identify, conceptualize, and quantify, their relationship with the propensity score, as encapsulated by ϕ, is very complex and varied from one domain to another. Therefore, we propose to use a neural network as the universal function approximator for ϕ.

Usage of Graph Features. We define the approximator for ϕ as:

$$\hat{\phi}(u, v) = \sigma(\text{MLP}([d_u; d_v; f(u); f(v); g(u, v)])) \tag{9}$$

where σ is a non-linearity function such as sigmoid; the degree features and the embeddings of u, v are concatenated as input to the MLP; $g(u, v) = g^+(f(u), f(v))$, see Sec.3.1. The output $\hat{\phi}(u, v)$ is directly plugged into Eq. 25 so that $\hat{\phi}$ is co-trained with f in an end-to-end fashion. The role of $\hat{\phi}$ in Eq. 25 is very similar to that of the attention mechanism (Veličković et al., 2018): both are a plugged-in module that learns to reweigh samples (tokens) alongside primary training; the goal of both modules is to approximate an underlying, physically motivated function.

Regularization. $\hat{\phi}$ can be further regulated to align its behavior with the desired properties of ϕ. Besides the unit-range output enforced by σ, we also consider its first moment and consistency:

- First moment: $\mathbb{E}_{u,v\sim p^+}[\phi_{u,v}] = c$;
- Consistency: $g(u, v_1) > g(u, v2) \Leftrightarrow \phi(u, v_1) > \phi(u, v_2), \quad \forall u, v_1, v_2 \in V$.

The first moment constraint restricts the numerical scale of ϕ, *i.e.* to be centered around c by expectation - see its proof in Appendix C.5. We softly enforce this by mean-squared loss $(\hat{\phi}(u, v) - c)^2$. The consistency constraint is based on the probabilistic gap theory (He et al., 2018; Gerych et al., 2022), which states the general order-preserving property between the propensity function and the decision function. We implement it by list-wise ranking loss ListMLE (Xia et al., 2008), computed between the two length-$(k + 1)$ lists, $(g(u, v^+), g(u, v_1^-), ..., g(u, v_K^-))$ and $(\hat{\phi}(u, v^+), \hat{\phi}(u, v_1^-), ..., \hat{\phi}(u, v_K^-))$.

Pseudocode. The full algorithm is presented as Algorithm 1 in Appendix.

Complexity. Despite being an intricate framework, our negative sampling has good time complexity due to importance sampling *i.e.* line 9 in Algorithm 1, which essentially reduces per-step complexity from $O(|V|)$ to $O(k)$. In fact, the only place that our negative sampling introduces time overhead is

at learning the propensity function, compared with the simplest uniform negative sampling. However, the computational cost of the additional MLP usually just a fraction of the main backbone model to learn. Our other debiasing steps involves computation of only some extra constants, as can be both seen from our pseudocode and validated by empirical results in Figure 3.

6 EXPERIMENT

6.1 EXPERIMENTAL SETUP

Dataset & Tasks. We primarily consider graph-based recommendations framed as link prediction problem, using three popular benchmarks whose sizes are among the largest in similar works: MovieLens (Ding et al., 2020), Pinterest (Ding et al., 2020; Geng et al., 2015), and LastFM Wang et al. (2019). A statistical overview of them can be found in Appendix E.1.

Baselines. We compare with 10 baseline methods covering both classical and state-of-the-art methods: DNS (Shi et al., 2023), AHNS (Lai et al., 2024), SENSEI Yan et al. (2024), MCNS (Yang et al., 2020b), NMNR Wang et al. (2018), IRGAN Wang et al. (2017), in-batch negatives with LogQ correction (Chen et al., 2020), MixGCF (Huang et al., 2021), uniformly random sampling (RNS), and popularity-based sampling (PNS) using node degrees.

Base Models. The choice of base learning model f is independent from the sampling. We consider two popular models: (1) a node embedding based model (Rendle et al., 2012), essentially performing matrix factorization and preceding many classical embedding methods such as deepwalk (Perozzi et al., 2014), and (2) LightGCN (He et al., 2020), a popular GNN model for recommendation.

Other Configurations. More details about data preprocessing and tuning are in Appendix E.2.

6.2 ANALYSIS OF PERFORMANCE

Table 1 shows that our method consistently outperforms the baselines. Two factors contribute to our better performance. First, we are the only method whose derivations are provably associated with the ground-truth negative distribution, while no baseline utilizes ground-truth negative distribution. Second, no baseline considers observation bias in sampling, resulting in overestimation of an edge's likelihood to be negative and deterioration of training data quality. In comparison, our method carefully mitigates these biases under a learnable framework.

DNS and AHNS are among the strongest baselines. While both baselines proactively probe for hard negatives, our method does *not* adopt that as first principle. It just happens so that the best form of sampling we derived aligns with the hard sampling strategy.

Ablation Study. We conduct various ablations to verify the effectiveness of the several key components in our sampling method. The results are reported in Table 2 and analyzed in Appendix E.5. The performance drop in each ablation demonstrates necessity of the designed component.

6.3 NEGATIVE SAMPLING FOR CONTRASTIVE LEARNING

Our method is *not* designed for *all* graph tasks. However, for comprehensiveness of study we also evaluate it in contrastive-learning-based node classification, using two classic frameworks as base models: GRACE Zhu et al. (2020) and GCE Zhu et al. (2021). We align with the original works on metrics used and rounding of reported numbers. Table 3 shows the result.

We see that our method not only performs well in general, but in some cases it even outperforms the original paper that uses full contrastive loss without any sampling - a phenomenon unobserved on link-based tasks. We conjecture that our debiasing and hard negative selection help the model learn more helpful structural information for node classification - see more discussion in Appendix E.4.

6.4 FURTHER ANALYSIS

We further study the numerical behaviors of all sampling methods on MovieLens dataset.

Convergence. Fig. 2 (a) plots the training loss curves for all sampling methods. Our method, despite also being a hard negative sampling method with a MLP module to train, converges relatively fast — mainly because it is derived from the ground up to strictly approximate the log-softmax loss $R_2(f)$ in an unbiased manner. In comparison, no heuristics-based method approximates a concrete loss term. Note that the absolute loss values are not directly comparable since their expressions vary.

Sensitivity to Hyperparameters. We investigate two core hyperparameters: positive class prior π^+, and negative sampling number k. The results are shown in Figs.2(b) - (c). Fig.2(b) shows a sweet

region for π^+. Note our theories suggest that we can directly observe (estimate) $P(s = 1)$ from data in an unbiased manner; this value is ~ 0.0027 for MovieLens, according to the plot. Since the positive class prior is $\pi^+ = P(y = 1)$, the ratio $P(s = 1)/\pi^+$ stands for the observation rate which is always below 1 in practice due to various biases. The sweet region of π^+ indicates that real observation rate for MovieLens might be $\sim 20\% - 35\%$, though unfortunately it is impossible to directly verify this.

We further study the effect of negative sampling number k in Figure 2 (c). The result is consistent with our expectation that the performance quickly plateaus as k increases.

Time Complexity. Fig.3(a) plots the epoch times, placing our method in middle range. Among the baselines, DNS and PNS run relatively slowly due to expansive search and lack of a localized sampling algorithm. We also found that most localized sampling methods run faster, whose sampling only accounts for a small fraction of total time. Fig.3(b) - (d) plot asymptotic times, showing a desirable near-linear trend which justifies our discussions on our method's complexity in Sec.5.

Table 1: Performance on real-world datasets measured by Recall@20 (%). Results on NDCG are reported in Appendix E.3. MovieLens and Pinterest have data splits to simulate biased observation, and LastFM to simulate unbiased observation. * means the best performance; ** means statistical significance; underline means the second best result. Baseline MixGCF is designed for GNN base models only.

Negative Sampling	Base Model: Node Embedding (Rendle et al., 2012)			Base Model: LightGCN (He et al., 2020)		
	MovieLens	Pinterest	LastFM	MovieLens	Pinterest	LastFM
RNS	7.11 ± 0.16	7.50 ± 0.19	5.97 ± 0.13	8.94 ± 0.11	7.98 ± 0.25	6.39 ± 0.10
PNS	8.41 ± 0.10	8.42 ± 0.05	6.64 ± 0.11	7.58 ± 0.05	8.67 ± 0.05	7.10 ± 0.09
LogQ(Chen et al., 2020)	9.97 ± 0.10	8.14 ± 0.09	4.74 ± 0.15	11.55 ± 0.20	11.53 ± 0.05	5.53 ± 0.11
DNS (Shi et al., 2023)	12.42 ± 0.09	8.56 ± 0.06	7.37 ± 0.21	11.89 ± 0.08	10.77 ± 0.13	7.26 ± 0.12
SENSEI Yan et al. (2024)	10.19 ± 0.06	9.01 ± 0.10	5.13 ± 0.25	8.10 ± 0.11	10.78 ± 0.14	5.44 ± 0.21
MCNS Yang et al. (2020a)	8.53 ± 0.08	8.68 ± 0.17	6.65 ± 0.20	7.22 ± 0.05	8.85 ± 0.11	6.88 ± 0.12
AHNS (Lai et al., 2024)	12.08 ± 0.09	10.84 ± 0.09	7.01 ± 0.10	12.73 ± 0.10	11.35 ± 0.09	6.90 ± 0.13
IRGAN (Wang et al., 2017)	10.57 ± 0.12	9.11 ± 0.09	6.86 ± 0.09	10.23 ± 0.25	9.79 ± 0.26	7.13 ± 0.11
NMRN Wang et al. (2018)	8.76 ± 0.19	7.67 ± 0.11	6.87 ± 0.11	10.22 ± 0.05	7.62 ± 0.06	7.12 ± 0.14
MixGCF Huang et al. (2021)	-	-	-	12.60 ± 0.07	10.25 ± 0.08	5.63 ± 0.07
Our Method	**12.77 ± 0.13****	**12.59 ± 0.12****	**7.38 ± 0.11***	**17.41 ± 0.19****	**17.61 ± 0.20****	**7.39 ± 0.12****

Table 2: Ablations of different components of our method demonstrate their necessity.

Ablation	MovieLens	Pinterest
(∗) Full method (control group)	12.77 ± 0.13	12.59 ± 0.12
(A) Removing adaptive reweight	7.60 ± 0.15	6.54 ± 0.20
(B) Constant propensity score	10.85 ± 0.19	9.38 ± 0.18
(C) Propensity w/o regularization	10.46 ± 0.40	8.99 ± 0.51
(D) Uninformative class prior	11.19 ± 0.11	9.39 ± 0.15
(E) Sampling for unbiased case	10.35 ± 0.18	8.01 ± 0.15

Table 3: Negative sampling applied to graph contrastive learning methods.

Method	GRACE Zhu et al. (2020)		GCE Zhu et al. (2021)	
	Cora	CiteSeer	Wiki-CS	Amazon-Photo
Full (control)	83.3 ± 0.4	72.1 ± 0.5	78.30 ± 0.00	92.49 ± 0.09
RNS	82.6 ± 1.0	70.4 ± 1.0	76.82 ± 0.03	90.46 ± 0.14
PNS	82.1 ± 0.8	71.3 ± 0.5	77.03 ± 0.05	*not converged*
DNS	81.3 ± 0.3	70.8 ± 0.6	77.78 ± 0.03*	92.91 ± 0.10*
AHNS	82.9 ± 0.4	71.0 ± 0.5	77.24 ± 0.07	90.98 ± 0.14
Our Method	83.6± 0.6	71.6± 0.8	77.41± 0.04	91.94± 0.15

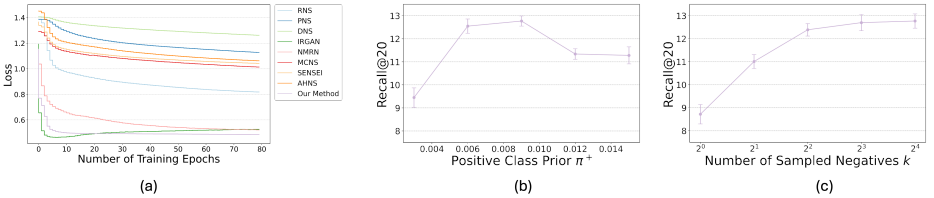

Figure 2: Numerical behaviors of sampling methods: (a) convergence curves of training losses; (b) model performance affected by positive class prior π^+; (c) model performance affected by the negative sampling number k. The experiments are done on MovieLens.

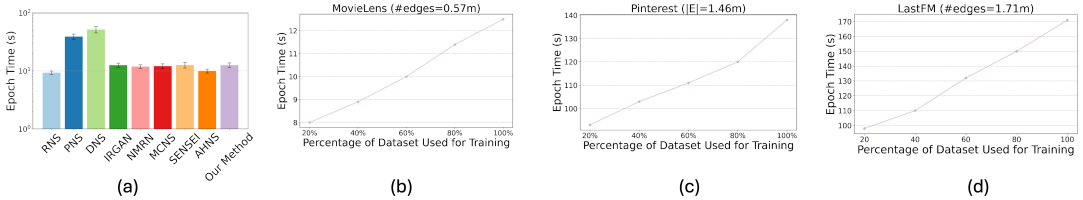

Figure 3: Investigating time complexity of our method: (a) Epoch times compared with other baselines on MovieLens; (b)-(d) Asymptotic time consumption tests on all datasets.

7 CONCLUSION

In this paper, we introduce a new paradigm for designing negative sampling for graph-based recommendations. The main novelty is that the derivation is based on first principles and leverages graph topology for learning-based debiasing of biased observations. Experiments show advantages over existing approaches, validating our theories and offering possibilities for better sampling algorithms.

Ethics Statement

This work does not raise specific ethical concerns. Our study focuses on negative sampling strategies in graph-based recommendation and link prediction, and does not involve human subjects, sensitive personal data, or applications with immediate societal risks. We do not foresee any direct ethical implications, nor do we address topics such as fairness, bias, or privacy beyond the standard scope of recommendation research. All experiments are conducted on publicly available benchmark datasets, and our results are reported in aggregate without identifying any individual users or entities.

Reproducibility Statement

This work makes sveral efforts to ensure reproducibility. Our code and dataset can be downloaded at `https://anonymous.4open.science/r/NS-Graph-Rec/`. Pseudocode is provided as Algorithm 1. Implementations and configuration details have been discussed in both main text and Appendix E.2.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pp. 137–146, 2018.

Chong Chen, Weizhi Ma, Min Zhang, Chenyang Wang, Yiqun Liu, and Shaoping Ma. Revisiting negative sampling vs. non-sampling in implicit recommendation. *ACM Transactions on Information Systems*, 41(1):1–25, 2023a.

Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. Bias and debias in recommender system: A survey and future directions. *ACM Transactions on Information Systems*, 41(3):1–39, 2023b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Jingtao Ding, Yuhan Quan, Quanming Yao, Yong Li, and Depeng Jin. Simplify and robustify negative sampling for implicit collaborative filtering. *Advances in Neural Information Processing Systems*, 33:1094–1105, 2020.

Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. Learning image and user features for recommendation in social networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 4274–4282, 2015.

Walter Gerych, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, and Elke Rundensteiner. Recovering the propensity score from biased positive unlabeled data. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 6694–6702, 2022.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of machine learning research*, 13(2), 2012.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Fengxiang He, Tongliang Liu, Geoffrey I Webb, and Dacheng Tao. Instance-dependent pu learning by bayesian optimal relabeling. *arXiv preprint arXiv:1808.02180*, 2018.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

Tinglin Huang, Yuxiao Dong, Ming Ding, Zhen Yang, Wenzheng Feng, Xinyu Wang, and Jie Tang. Mixgcf: An improved training method for graph neural network-based recommender systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 665–674, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Riwei Lai, Rui Chen, Qilong Han, Chi Zhang, and Li Chen. Adaptive hardness negative sampling for collaborative filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8645–8652, 2024.

Haokai Ma, Ruobing Xie, Lei Meng, Fuli Feng, Xiaoyu Du, Xingwu Sun, Zhanhui Kang, and Xiangxu Meng. Negative sampling in recommendation: A survey and future directions. *arXiv preprint arXiv:2409.07237*, 2024.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

Indraneel Mukherjee, Cynthia Rudin, and Robert E Schapire. The rate of convergence of adaboost. In *Proceedings of the 24th Annual Conference on Learning Theory*, pp. 537–558. JMLR Workshop and Conference Proceedings, 2011.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Aleksandr Vladimirovich Petrov and Craig Macdonald. gsasrec: Reducing overconfidence in sequential recommendation trained with negative sampling. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pp. 116–128, 2023.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

CP Robert. Monte carlo statistical methods, 1999.

Joshua David Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2021.

Evgenia Rusak, Patrik Reizinger, Attila Juhos, Oliver Bringmann, Roland S Zimmermann, and Wieland Brendel. Infonce: Identifying the gap between theory and practice. *arXiv preprint arXiv:2407.00143*, 2024.

Wentao Shi, Jiawei Chen, Fuli Feng, Jizhi Zhang, Junkang Wu, Chongming Gao, and Xiangnan He. On the theories behind hard negative sampling for recommendation. In *Proceedings of the ACM Web Conference 2023*, pp. 812–822, 2023.

Jie Shuai, Kun Zhang, Le Wu, Peijie Sun, Richang Hong, Meng Wang, and Yong Li. A review-aware graph contrastive learning framework for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pp. 1283–1293, 2022.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 776–794. Springer, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 515–524, 2017.

Qinyong Wang, Hongzhi Yin, Zhiting Hu, Defu Lian, Hao Wang, and Zi Huang. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2467–2475, 2018.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 950–958, 2019.

Yanbang Wang, Karl Hallgren, and Jonathan Larson. A graph-based framework for reducing false positives in authentication alerts in security systems. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 274–283, 2024.

Yingheng Wang, Yaosen Min, Xin Chen, and Ji Wu. Multi-view graph contrastive representation learning for drug-drug interaction prediction. In *Proceedings of the web conference 2021*, pp. 2921–2933, 2021.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. Pmlr, 2019.

Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 726–735, 2021.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pp. 1192–1199, 2008.

Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. Reconciling competing sampling strategies of network embedding. *Advances in Neural Information Processing Systems*, 36, 2024.

Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion proceedings of the web conference 2020*, pp. 441–447, 2020a.

Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1666–1676, 2020b.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983, 2018.

Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pp. 1294–1303, 2022.

Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pp. 785–788, 2013.

Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. Causal intervention for leveraging popularity bias in recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 11–20, 2021.

Han Zhao, Xu Yang, Zhenru Wang, Erkun Yang, and Cheng Deng. Graph debiased contrastive learning with joint representation clustering. In *IJCAI*, pp. 3434–3440, 2021.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the web conference 2021*, pp. 2069–2080, 2021.

Appendix

A Implicit Data Assumptions of Existing Methods

Should we consider negative sampling strategies to be primarily a matter of human design, or does a ground-truth distribution of negative samples p^- exist objectively and independently of our choices (and further that its exact form in the real world cannot be easily articulated by "principles" due to nature's complexity)?

For this question, an interesting observation is that many of the existing methods have implicitly taken taken an ambiguous or mixed stance.

To see this, notice that many previous works for *negative* sampling have left *positive* sampling untouched. This essentially respects the existence of a ground-truth positive distribution in nature, from which the observed data (positive edges) get sampled unbiasedly. However, respecting the observed ground-truth positive distribution is equivalent to respecting its corresponding ground-truth negative distribution (by simply taking complement). This is because any graph in nature can be simultaneously viewed as a collection of positive edges sampled from the positive distribution, or a collection of negative edges sampled from the negative distribution. However, in the meantime those existing methods are proposing new negative distributions.

This seeming self-contradiction can only be explained when we consider the existing methods to have respected the existence of p^-, but in meantime be proposing a negative distribution different from their respected ground-truth for other good learning properties such as potentially faster empirical convergence. However, this advantage is at the compromise of unbiased estimation, and often does not have theoretical guarantee.

B Further Clarification on the Scope of this Work

Target tasks. We do *not* propose the negative sampling method as a general strategy for all graph tasks. In fact, none of the existing literature on negative sampling motivates their design as a one-fit-for-all general strategy. It is also unlikely that there exists an optimal sampling strategy for all graph tasks which have different downstream objectives to optimize for. Like most of the existing works on negative sampling, our problem formulation in Section 3.1 targets the problem of recommendation and link prediction.

Incremental or Streaming Updates. Our method naturally supports incremental or streaming updates with its stream-based algorithmic design – as shown in Algorithm 1 (line 5): it iterates on a sequence of edges, rather than an offline static graph as a whole. Therefore, it naturally allows for online learning, as well as random split of new edge streams (and thus A/B testing).

Our theories and methodology are also agnostic to whether the underlying graph is static or evolving. Although we start with a canonical definition of graph in Sec. 3.1, our entire discussion has switched into a probabilistic edge space since Sec. 3.2. Therefore, our method holds as long as incremental updates do not introduce further distribution shift, a topic outside of the scope of this work.

C Proofs and Theoretical Discussion

C.1 Proof of Proposition 3.1

$$J = \sum_{(u,v)\in E,\, v_1^-,\ldots,v_k^- \sim q^-(v)} \left[L^+(u,v) + \frac{\tau}{k} \sum_{i=1}^{k} L^-(u, v_i^-) \right] \tag{10}$$

$$\simeq \mathbb{E}_{(u,v)\sim p^+} \left[L^+(u,v) + \tau \mathbb{E}_{v^-\sim q^-(v)} L^-(u,v^-) \right] \tag{11}$$

$$= \mathbb{E}_{u\sim p^+} \left[\mathbb{E}_{v\sim p^+(\cdot|u)} \left[L^+(u,v) + \mathbb{E}_{v^-\sim q^-(v)} L^-(u,v^-) \right] \right] \tag{12}$$

$$= \mathbb{E}_{u\sim p^+} \left[\mathbb{E}_{v\sim p^+(\cdot|u)} L^+(u,v) + \mathbb{E}_{v\sim p^+(\cdot|u)} \left[\mathbb{E}_{v^-\sim q^-(v)} \left[L^-(u,v^-) \right] \right] \right] \tag{13}$$

$$= \mathbb{E}_{u\sim p^+} \left[\mathbb{E}_{v\sim p^+(\cdot|u)} L^+(u,v) + \mathbb{E}_{v^-\sim q^-(v)} L^-(u,v^-) \right] \tag{14}$$

C.2 Generalized Softmax

The full form of the generalized softmax is: for a selected edge (u, v),

$$p_f(v|u) = \frac{e^{g(f(x_u),f(x_v))} + \tau p^-(v|u)e^{g(f(x_u),f(x_v))}}{e^{g(f(x_u),f(x_v))} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g(f(x_u),f(x_{v^-}))}} \tag{15}$$

$$p_f(v'|u) = \frac{\tau p^-(v'|u)e^{g(f(x_u),f(x_{v'}))}}{e^{g(f(x_u),f(x_{v'}))} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g(f(x_u),f(x_{v^-}))}}, \text{ for all } v' \neq v \tag{16}$$

This ensures that the probability distribution under the generalized softmax term in Eq. 5 is proper, because the denominator is not symmetric for all $v \in V$, *i.e.* the selected edge (u, v) is treated differently than all other edge (u, v') where $v' \neq v$. In practice since the support of $p^-(\cdot|u)$ usually does not encompass the positive node v in the selected positive edge (u, v), *i.e.* $p^-(v|u) = 0$, Eq. 15 degrades into Eq. 5.

C.3 Proof of Proposition 4.1

Denote $g(f(x_u), f(x_v))$ by g_{uv}

$$\nabla_\theta R_2(f) = \nabla_\theta \, \mathbb{E}_{(u,v)\sim p^+}[-\log \frac{e^{g_{uv}}}{e^{g_{uv}} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g_{uv^-}}}] \tag{17}$$

$$= \mathbb{E}_{(u,v)\sim p^+}[-\nabla_\theta g_{uv} + \nabla_\theta \log(e^{g_{uv}} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g_{uv^-}})] \tag{18}$$

$$= \mathbb{E}_{(u,v)\sim p^+}[-\nabla_\theta g_{uv} + \frac{e^{g_{uv}} \nabla_\theta g_{uv} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g_{uv^-}} \nabla_\theta g_{uv^-}}{e^{g_{uv}} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g_{uv^-}}}] \tag{19}$$

$$= \mathbb{E}_{(u,v)\sim p^+}[-\nabla_\theta g_{uv} + \frac{e^{g_{uv}} \nabla_\theta g_{uv}}{e^{g_{uv}} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g_{uv^-}}} + \frac{\tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g_{uv^-}} \nabla_\theta g_{uv^-}}{e^{g_{uv}} + \tau \mathbb{E}_{v^- \sim p^-(\cdot|u)} e^{g_{uv^-}}}] \tag{20}$$

$$= \mathbb{E}_{(u,v)\sim p^+}[-\nabla_\theta g_{uv} + \mathbb{E}_{v^- \sim p^-(\cdot|u)}[p_f(v^-|u) \nabla_\theta g_{uv^-}]] \tag{21}$$

$$= \mathbb{E}_{u\sim p^+}[\mathbb{E}_{v\sim p^+(\cdot|u)}[\nabla_\theta(-g_{uv})] + \mathbb{E}_{v^- \sim p^-(\cdot|u)}[p_f(v^-|u) \nabla_\theta g_{uv^-}]] \tag{22}$$

Notice that the first summation term is exactly positive sampling, and the second summation term is exactly negative sampling. Substitute $g^+ = -g_{uv} = -g(f(x_u), f(x_v))$, $g^- = z_u g(f(x_u), f(x_v))$ into the equation above:

$$\nabla_\theta \, R_2(f) = \mathbb{E}_{u\sim p^+}[\mathbb{E}_{v\sim p^+(\cdot|u)}[\nabla_\theta L^+(u, v)] + \mathbb{E}_{v^- \sim p^-(\cdot|u)}[\frac{1}{z_u} p_f(v^-|u) \nabla_\theta L^-(u, v^-)]] \tag{23}$$

where the partition $z_u = \mathbb{E}_{v\sim p^-(\cdot|u)}[p_f(v|u)]$. Meanwhile, from the definition of $R_1(f)$ in Eq. 4:

$$\nabla_\theta \, R_1(f) = \mathbb{E}_{u\sim p^+}[\mathbb{E}_{v\sim p^+(\cdot|u)}[\nabla_\theta L^+(u, v)] + \mathbb{E}_{v^- \sim q^-}[\nabla_\theta L^-(u, v^-)]] \tag{24}$$

Therefore,

$\nabla_\theta R_1(f) \equiv \nabla_\theta R_2(f)$ if and only if $\forall u, v \in V, \;\; q = \tau p_f(v|u) p^-(v|u)/z_u \propto p_f(v|u) p^-(v|u)$.

C.4 Proof of Theorem 5.1

Define $p = P(e)$ to be the node pair prior. We first have the following propositions.

Proposition C.1. $p^+ = \frac{c}{\phi_{u,v}} \hat{p}^+$.

Proof. $\hat{p}^+ = P(e|s = 1) = P(e|y = 1)P(s = 1|y = 1, e)/P(s = 1|y = 1) = p^+ \frac{\phi(e)}{c}$. □

Proposition C.2. $p^- = \frac{1}{\pi^-}(p - \pi^+ p^+)$.

Proof. $p = P(e) = P(e|y = 1)P(y = 1) + P(e|y = 0)P(y = 0) = \pi^+ p^+ + \pi^- p^-$. □

Proposition C.3. $p = \pi^+ c\hat{p}^+ + (1-\pi^+ c)\hat{p}^-$.

Proof. $P(s=1) = P(s=1, y=1) = P(s=1|y=1)P(y=1) = \pi^+ c$, so $p = P(e) = P(e|s=1)P(s=1) + P(e|s=0)P(s=0) = \pi^+ c\hat{p}^+ + (1-\pi^+ c)\hat{p}^-$. □

Based on the above propositions, we have

$$R(f,u) = \mathbb{E}_{v\sim p^+}[L^+] + \mathbb{E}_{v\sim p^-}[p_f(v)L^-] \tag{25}$$

$$= \mathbb{E}_{v\sim p^+}[L^+] + \mathbb{E}_{v\sim(\frac{1}{\pi^-}(p-\pi^+p^+))}[p_f(v)L^-] \tag{26}$$

$$= \mathbb{E}_{v\sim p^+}[L^+ - \frac{\pi^+}{\pi^-}p_f(v)L^-] + \mathbb{E}_{v\sim p}[\frac{1}{\pi^-}p_f(v)L^-] \tag{27}$$

$$= \mathbb{E}_{v\sim \frac{c}{\phi_{u,v}}\hat{p}^+}[(L^+ - \frac{\pi^+}{\pi^-}p_f(v)L^-)] + \mathbb{E}_{v\sim(\pi^+c\hat{p}^+ + (1-\pi^+c)\hat{p}^-)}[\frac{1}{\pi^-}p_f(v)L^-] \tag{28}$$

$$= \mathbb{E}_{v\sim\hat{p}^+}[\frac{c}{\phi_{u,v}}L^+] - \mathbb{E}_{v\sim\hat{p}^+}[\frac{\pi^+c}{\pi^-\phi_{u,v}}p_f(v)L^-] + \mathbb{E}_{v\sim\hat{p}^+}[\frac{\pi^+c^2}{\pi^-\phi_{u,v}}p_f(v)L^-] + \mathbb{E}_{v\sim\hat{p}^-}[\frac{(1-\pi^+c)c}{\pi^-\phi_{u,v}}p_f(v)L^-] \tag{29}$$

$$= \mathbb{E}_{v\sim\hat{p}^+}[\frac{c}{\phi_{u,v}}L^+] + \mathbb{E}_{v\sim\hat{p}^+}[\frac{\pi^+c(1-c)}{\pi^-\phi_{u,v}}p_f(v)(-L^-)] + \mathbb{E}_{v\sim\hat{p}^-}[\frac{(1-\pi^+c)c}{\pi^-\phi_{u,v}}p_f(v)L^-] \tag{30}$$

Substitute definitions of $\beta_1, \beta_2, q^+, q^-$ into Eq. 5.1, we will see it exactly equals $R_2(f)$ in Theorem 5.1.

C.5 Proof of the First-moment Constraint in Section 5.3

$\mathbb{E}_{u,v\sim p^+}[\phi_{u,v}] = \mathbb{E}_{u,v\sim p^+}[P(s=1|y=1, e=(u,v)] = \sum_{u,v\in V}[P(e=(u,v)|y=1)P(s=1|y=1, e=(u,v)] = \sum_{u,v\in V}[P(s=1, e=(u,v)|y=1)] = \sum_e[P(s=1, e=(u,v)|y=1)] = P(s=1|y=1)x = c$

C.6 Proof of the Consistency Constraint in Section 5.3

The probabilistic gap theory (He et al., 2018; Gerych et al., 2022) essentially states that $\phi(e) = h(P(y=1|e) - P(y=0|e))$, $h'(t) > 0$. Therefore, $\phi(e) = h(2g(e)-1)$ and $\frac{d\phi}{dg}|_{t=e} = 2h'(g(e)) > 0$. Equivalently, $\phi(e_1) > \phi(e_1) \Leftrightarrow g(e_1) > g(e_1)$.

D Pseudocode

Pseudocode for the full algorithm can be found in Algorithm 1.

E Experiment

E.1 Dataset

The sources for the datasets used in this paper are:

- MovieLens (Harper & Konstan, 2015): `https://grouplens.org/datasets/MovieLens/100k/`.
- Pinterest (Geng et al., 2015): `https://github.com/edervishaj/pinterest-recsys-dataset?tab=readme-ov-file`
- LastFM: (Wang et al., 2019): `https://github.com/xiangwang1223/knowledge_graph_attention_network`

All datasets are public, and have been in included in the submitted codes.

Algorithm 1: Proposed Debiased Negative Sampling

Input : graph $G = (V, E)$, where observed positive edges $E = \{(u, v)\}$; graph encoder f with parameters θ to train; number of negative samples k; positive class prior π^+; number of training epochs T; similarity function g.

Compute constants and node prior $\pi^- \leftarrow 1 - \pi^+$, $c \leftarrow \frac{2|E|}{|V|(|V|-1)\pi^+}$, $p(v) \leftarrow \frac{1}{|V|}, \forall v \in V$
Initialize propensity score function ϕ as an MLP with parameters θ_ϕ
for $t = 1, 2, \cdots, T$ **do**
  Initialize loss $\mathcal{L} = 0$
  for *each* (u, v) *pair* **do**
    Retrieve u's adjacency list $\mathcal{N}(u) \leftarrow \{v' | (u, v') \in E\}$
    Compute positive distribution $\hat{p}^+(v') \leftarrow \frac{1}{|\mathcal{N}(u)|} \sum_{v'' \in \mathcal{N}(u)} \delta_{v''}, \forall v' \in V$
    Compute negative distribution $\hat{p}^-(v') \leftarrow (p(v') - \hat{p}^+(v')\pi^+ c)/(1 - \pi^+ c), \forall v' \in V$
    Sample negative nodes $v_1^-, ..., v_k^- \sim \hat{p}^-$ via importance sampling
    Compute learnable propensity score $\phi_{u,v}$ by Eq. (11)
    for *each* $v^- \in \{v_1^-, ..., v_k^-\}$ **do**
      Compute generalized softmax probability $p_f(v^-)$ by Eq. (5)
      Compute propensity score ϕ_{u,v^-} and its regularization penalty L_R by Eq. (11)
      Compute loss terms
        $L_v^+ \leftarrow g(f(u), f(v)); L_v^- \leftarrow -g(f(u), f(v)); L_{v^-}^- \leftarrow -g(f(u), f(v^-))$
      Aggregate reweighed loss terms
        $\mathcal{L} \leftarrow \mathcal{L} + \left[\frac{c}{\phi_{u,v}} L_v^+ + \frac{\pi^+ c(1-c)}{\pi^- \phi_{u,v}} p_f(v)(-L_v^-)\right] + \left[\frac{(1-\pi^+ c)c}{\pi^- \phi_{u,v^-}} p_f(v^-) L_{v^-}^-\right] + L_R$
  Update parameters θ, θ_ϕ by descending the gradients $\nabla_{\theta,\theta_\phi} \mathcal{L}$

Table 4: Statistics of datasets

Dataset	Training Edges	Validation Edges	Testing Edges	Users	Items
MovieLens (Ding et al., 2020)	563,112	6,028	6,028	6,028	3,533
Pinterest (Ding et al., 2020)	1,355,844	55,186	55,186	55,186	9,916
LastFM (Wang et al., 2019)	1,198,808	256,887	256,888	23,566	28,126

E.2 Data Split & Configurations

Data Split. Special care is needed when splitting the edge set into training, validation and testing. The most ideal setup should ensure that $E_{\text{train}}, E_{\text{val}} \sim \hat{p}^+$ and $E_{\text{test}} \sim p^+$.

For the unbiased scenario, we preprocess LastFM dataset to create data splits that simulate the case, which further requires $p^+ = \hat{p}^+$. Therefore the data split is done completely at random, *i.e.* $70\%, 15\%, 15\%$ for training, validation, and testing, respectively.

For the biased scenario, we use MovieLens and Pinterest datasets. Since we have no direct access to p^+. we split the edges in temporal order by following the commonly used "leave-one-out" strategy as in He et al. (2017); Rendle et al. (2012): leaving the most recent and the second most recent edge for each node as testing and validation set, respectively. We believe that using temporal splits is one of the most natural ways to introduce realistic bias to the test.

Configurations. The configurations for base models are to be fixed. In this paper, we use Adam optimizer, learning rate $1e-4$, L2 regularization $1e-5$, hidden embedding size 64, mini-batch size 1024, with early stopping. The GNN model used is a two-layered LightGCN. We use lower-bound clipping at $\epsilon = 5e-2$ and temperature scaling $T = 1.1$ to further regularize ϕ's output logits. We set $\tau = k = 8$ for both our method and baselines to ensure fair comparison. We further tune the class prior π_+ ranged from 1.5 times to 20 times of the directly observable $P(S = 1)$; we set g^- as the dot product and $g^+ = -g^-$. For hyperparameters related to the baseline sampling methods, we follow their reported settings for tuning. All experiments are run on an Nvidia V100 GPU.

E.3 PERFORMANCE BY NDCG

See Table 5.

Table 5: Performance of different sampling methods on real-world datasets, measured by NDCG@20 (%). * means the best performance; ** means the best performance with significance; underline means the second best result.

Negative Sampling	Node Embedding (Rendle et al., 2012)			LightGCN (He et al., 2020)		
	MovieLens	Pinterest	LastFM	MovieLens	Pinterest	LastFM
RNS	2.82 ± 0.10	3.01 ± 0.16	2.59 ± 0.18	4.00 ± 0.12	2.97 ± 0.20	2.98 ± 0.11
PNS	3.57 ± 0.11	3.35 ± 0.06	3.02 ± 0.14	3.21 ± 0.04	3.70 ± 0.04	1.59 ± 0.07
DNS (Shi et al., 2023)	4.78 ± 0.07	4.44 ± 0.07	3.58 ± 0.16	4.92 ± 0.05	4.90 ± 0.09	3.33 ± 0.08
SENSEI Yan et al. (2024)	4.01 ± 0.06	4.20 ± 0.08	2.60 ± 0.23	3.54 ± 0.10	4.88 ± 0.11	2.04 ± 0.19
MCNS Yang et al. (2020a)	4.08 ± 0.06	4.22 ± 0.09	3.02 ± 0.12	3.17 ± 0.05	3.75 ± 0.08	2.73 ± 0.08
AHNS (Lai et al., 2024)	4.35 ± 0.06	4.66 ± 0.08	3.38 ± 0.10	5.90 ± 0.08	5.92 ± 0.08	3.38 ± 0.10
IRGAN (Wang et al., 2017)	4.12 ± 0.10	3.39 ± 0.10	3.27 ± 0.08	4.75 ± 0.16	4.04 ± 0.18	3.02 ± 0.08
NMRN Wang et al. (2018)	3.93 ± 0.11	2.76 ± 0.12	3.22 ± 0.08	4.50 ± 0.06	3.11 ± 0.05	3.14 ± 0.10
Our Method	**5.01 ± 0.09****	**5.06 ± 0.09****	**3.70 ± 0.12***	**7.21 ± 0.13****	**7.28 ± 0.11****	**3.52 ± 0.10****

E.4 MORE DISCUSSION ON RESULTS FOR CONTRASTIVE LEARNING

First, we note that the method that uses all possible negatives essentially assumes that the best negative distribution to use is a uniform distribution over all observed negatives. However, according to our paper's analysis, this uniform distribution is most likely not the ground-truth negative distribution (i.e. the motivated in our paper), nor could it be the optimal negative distribution to sample from for training the model.

In comparison, our method is sampling from a carefully learned negative distribution, although it has a very low sampling number (k=8) that is incomparable to the full dataset. Therefore, we conjecture that our win benefits from that our learned negative distribution is closer to the optimal negative distribution, compared with the uniform distribution implicitly assumed by the method using full negatives. This closeness in distribution compensates for the disadvantage of using a lower sampling number. We thank the reviewer for give us an opportunity to elaborate on this interesting phenomenon.

E.5 ABLATION STUDIES

To further understand the contribution of each component, we conduct ablation studies on MovieLens and Pinterest datasets. Table 6 summarizes the different ablations and their results, and we have the following analysis.

Table 6: Ablations of different components of our method, tested on MovieLens and Pinterest. The metric is Recall@20 (%).

Ablation	**MovieLens**	**Pinterest**
(A) Removing adaptive reweight	7.60 ± 0.15	6.54 ± 0.20
(B) Constant propensity score	10.85 ± 0.19	9.38 ± 0.18
(C) Propensity w/o regularization	10.46 ± 0.40	8.99 ± 0.51
(D) Uninformative class prior	11.19 ± 0.11	9.39 ± 0.15
(E) Sampling for unbiased case	10.35 ± 0.18	8.01 ± 0.15
$(*)$ Full method (no ablation)	12.77 ± 0.13	12.59 ± 0.12

In ablation (A), we remove the adaptive reweighing component $p_f(v)$ which essentially functions as a filter for hard negatives. This results in most significant performance drop compared to all methods, which demonstrates its importance and supports our theoretical derivation in Proposition 4.1.

Ablations (B) and (C) study the propensity score estimation function $\phi_{u,v}$. In (B), we replace $\phi_{u,v}$ by its expectation over all possible (u, v)'s, which is the constant c defined in Theorem 5.1. In (C), we remove its regularization in training, essentially removing the constraints that regulate $\phi_{u,v}$ to

be numerically appropriate. Both ablations are shown to affect performance. we also observe (C)'s gap to be larger, which seems to suggest that a numerically well-behaved but inaccurate $\phi_{u,v}$ is more helpful than a numerically ill-behaved but presumably more accurate $\phi_{u,v}$ in debiasing.

Ablation (D) extends the sensitivity analysis in Section 6.4 by setting the hyperparameter of positive class prior π^+ to be 0.5. Again this ablation leads to performance drop, which helps support our theoretical derivations.

Finally, Ablation (E) examines the case where we over-simplify the data assumption — by applying our base sampling method derived for unbiased observations to address the biased case. The performance drop implies the importance of considering observation bias in negative sampling.

F Notation Table

Table 7: Summary of Notations in the Work

Notation	Description
Graph and Model	
$G = (V, E)$	The graph consisting of node set V and observed edge set E.
u, v	Nodes in the graph, where u typically denotes an anchor node.
x_u	Raw feature vector associated with node u.
$f_\theta(\cdot)$	Graph encoder function parameterized by θ.
$g(\cdot, \cdot)$	Similarity function (e.g., dot product) between two node embeddings.
L^+, L^-	Loss functions for positive and negative node pairs, respectively.
$R(f)$	Expected risk term for the encoder f.
Probabilistic Definitions	
$y \in \{0, 1\}$	Ground-truth class label indicator (1 for true edge, 0 otherwise).
$s \in \{0, 1\}$	Observation indicator variable (1 if observed, 0 otherwise).
p^+, p^-	True positive distribution $P(e\|y = 1)$, true negative distribution $P(e\|y = 0)$.
$\hat{p}^+, \hat{p}^-$	Observed positive distribution $P(e\|s = 1)$, observed negative distribution $P(e\|s = 0)$.
q^-	Proposal negative distribution used for sampling.
$p_f(v\|u)$	Generalized softmax probability of v given u based on current model output.
Bias and Propensity	
$\phi_{u,v}$	Propensity score $P(s = 1\|y = 1, e = (u, v))$; probability of observing a true positive edge.
π^+	Positive class prior probability $P(y = 1)$.
c	Observation rate $P(s = 1\|y = 1)$, assumed to be the expected value of ϕ.
β_1, β_2	Re-weighting constants derived for the unbiased risk estimator.

G The Use of Large Language Models

Large Language Models (LLMs) are minimally used in preparing this manuscript and should not be considered a substantial contributor. LLMs have only been used to adjust tables and figures layout, and to ensure grammatical correctness.