Algorithms and Evaluation for News Recommender Systems

September 2022

Kojiro Iizuka

Algorithms and Evaluation for News Recommender Systems

Kojiro Iizuka

Doctoral Program in Informatics
Degree Programs in Comprehensive Human Sciences
Graduate School of Comprehensive Human Sciences
University of Tsukuba

September 2022

Algorithms and Evaluation for News Recommender Systems

Kojiro Iizuka

News recommender systems are important for users to access the information they need from the vast information in daily life. At the same time, the recommender system is an essential source of revenue for news providers. In order to recommend useful news articles to users and generate revenue effectively, news recommender systems need to overcome a number of challenging issues that arise from the characteristics.

Recency is a characteristic of news recommender systems that the value of news articles decreases significantly over time. Due to this recency, news recommender systems need to deliver news immediately to users as it becomes available. In addition, news recommender systems have a high velocity in that new articles appear one after another at a rapid pace. This high velocity makes it difficult to perform and evaluate recommender algorithms since there is little user behavior data to perform or evaluate algorithms for new items. Another characteristic is *sponsored ranking*. News recommender systems have characteristics that mix organic news articles and ads as rankings to present them to users, and we call this ranking a sponsored ranking that includes ads sponsored by companies. It is important for a sponsored ranking to increase user satisfaction by delivering news content that is attractive to users and, at the same time, to ensure advertising revenue. This thesis addresses challenges that arise from the characteristics of recency, high-velocity, and sponsored ranking in the news recommendation as follows.

An Accurate and Efficient Embedding Method for News Recommendations

Online news recommender systems must deliver news articles immediately that users prefer as they become available. In order to build a system that satisfies this requirement, the similarity between news articles and users must be calculated in real-time. Recently, embedding-based recommender systems have become standard methods to achieve high accuracy by using machine learning models to represent items and users with vectors. The key to the real-time response using the embedding-based news recommender systems is a small embedding dimension. If the embedding dimension is large, the system can not respond in real-time since the cost of similarity calculation using embedding exceeds the computing resources. This thesis proposes

a hyperbolic embedding method that embeds recommendation items into vectors in a hyperbolic space, which can accurately capture the relationship between users and items even with a small dimension. In addition, we propose a strategy of successive model updating to learn user behavior data efficiently. It is challenging to characterize news articles using user behavior data for newly available news because there is little user behavior data. The idea of the proposed method is to update the trained model successively. Thanks to the idea of successive model updating, the training can start with, the training can start with the inherited features of the previously trained model, and higher accuracy can be achieved compared to training the model from scratch, with the same number of training epochs.

An Interleaving Method for Evaluating Post-click Metrics

The post-click user behavior is important in recommender systems since post-click user behavior indirectly expresses the user's satisfaction. Post-click user behavior in a news recommender system includes article reading behavior, which is the behavior after clicking on an article. If a user clicks on a news article and then stops reading the article within a few seconds, the data is considered as a signal that the clicked article was not desirable and the user was unsatisfied. Since news articles in the news recommender system appear one after another at a rapid pace, the post-click data for a single news article is limited. Therefore, when using post-click data for evaluation in a news recommender system, it is necessary to utilize a small amount of post-click data effectively. In this thesis, we propose a method to evaluate post-click metrics even with a small amount of post-click data. The proposed method aims to reduce variance in evaluation by using importance sampling, in which items with a small number of clicks are preferentially presented to users.

Evaluating the Effects of News Articles on Ad Consumption

The challenge in the news recommender systems is to generate effective ranking with organic items and ad items. The effect between news items and ad items should be carefully investigated to generate an effective ranking in the news recommender systems since one might influence the other and reduce the overall effectiveness of the ranking. This thesis evaluates how news articles affect users' ad consumption in sponsored ranking and shows the dependency between news and ad consumption. We first conducted a service log analysis and showed that sessions with high-quality news recommendations had more ad consumption than low-quality news articles recommendations. Based on this result, we hypothesized that high-quality news

recommendation leads to a high ad consumption rate. Thus, we conducted million-scale A/B testing to investigate the effect of high-quality articles on ad consumption. We presented high-quality articles at the top of group-A's ranking and presented both low-quality and high-quality articles to group-B. The A/B testing showed that the group-A's ad consumption, such as clicks, conversions, and sales, increased significantly while the number of article clicks decreased.

Contents

Contents **i**

1 Introduction **1**

1.1 Background 1

1.2 Characteristics of News Recommender Systems 2

1.3 Challenges of News Recommender Systems 4

1.4 Thesis Organization 6

2 Related Work **9**

2.1 Embedding-based Recommendation 9

2.1.1 Small Dimension Embeddings 9

2.1.2 Efficient Model Updating 10

2.2 Post-click Evaluation 11

2.2.1 Interleaving Methods 11

2.2.2 Variance Reduction 11

2.2.3 Off-policy Evaluation 12

2.2.4 Bandit Algorithms 12

2.3 Sponsored Ranking Evaluation 13

2.3.1 User Behavior Analysis 13

2.3.2 Optimization of Organic Content and Advertisements 14

3 An Accurate and Efficient Embedding Method for News Recommendations **15**

3.1 Introduction 15

3.2 Analysis 17

3.2.1 Notation 17

3.2.2 Datasets 18

3.2.3 Metrics 18
3.2.4 Results 19
3.3 Regularization 20
3.3.1 Hyperbolic Embeddings 20
3.3.2 Regularization 23
3.4 Dynamic Embedding 25
3.4.1 Problem Definition 25
3.4.2 Dynamic Model Update 26
3.5 Offline Experiment 27
3.5.1 Metrics 28
3.5.2 Accuracy Evaluation 30
3.5.3 Efficiency Evaluation 33
3.6 Online Experiment 35
3.6.1 Experimental Setup 36
3.6.2 Online Metrics 36
3.6.3 Results 37
3.7 Summary 37

4 An Interleaving Method for Evaluating Post-click Metrics 39
4.1 Introduction 39
4.2 Problem Setting 41
4.3 Methodology 42
4.3.1 Decomposition 43
4.3.2 Interleaving 45
4.3.3 Theoretical Justification 46
4.3.4 Stabilization Techniques 47
4.4 Experiment Setup 49
4.4.1 Simulation-based Settings 50
4.4.2 Real Service Settings 53
4.4.3 Parameter Estimation 55
4.4.4 Variance Prediction 55
4.4.5 Comparison Methods 56
4.5 Results and Discussion 57
4.5.1 Efficiency and Accuracy 57

4.5.2 Variance Prediction Technique 58
4.5.3 Error Correction Technique 59
4.6 Summary 60

5 Evaluating the Effects of News Articles on Ad Consumption 61
5.1 Introduction 61
5.2 Preliminary 62
5.2.1 Target Service 63
5.2.2 Article Quality 63
5.2.3 Metrics 64
5.3 Log Analysis 65
5.3.1 Data 65
5.3.2 Article Consumption 65
5.3.3 Ad Consumption 66
5.4 Online Experiment 67
5.4.1 Setting 67
5.4.2 Results 68
5.5 Summary 71

6 Conclusions 72
6.1 Summary 72
6.1.1 Technical Contributions 74
6.1.2 Social Contributions 74
6.2 Future Directions 74

Acknowledgements i

Bibliography ii

Publications xiii
Journal Articles xiii
Conference Papers xiii
Domestic Symposium xiv

Chapter 1

Introduction

1.1 Background

Recommender systems are essential for users to access the information they need from the vast amount available. Recommender systems became widely known through a feature in ACM Communications and have since been the subject of much research [70]. Following the definition of Melville, the goal of a recommender system is to generate meaningful recommendations to a collection of users for items or products that might interest them [54]. Recently, recommender systems are being used by many companies since users have more opportunities to access information daily with the development of internet technology and the spread of smartphones. For example, Amazon's recommender system helps users make decisions to purchase products by recommending related products to them. According to McKinsey's study, Amazon's sales through recommender systems account for 35% of its total sales [52]. In addition, YouTube's recommender system is equipped with a function that recommends videos suited to the user based on the history of videos the user has viewed, contributing to the improvement of the video viewing experience [25]. YouTube generates revenue by running advertisements before users play the videos they want to watch where improving the video viewing experience through the recommender system significantly impacts revenue. Thus, recommender systems play a role in connecting information and users. At the same time, they are an essential source of revenue for companies.

News recommender systems are one of the most common ways by which users obtain information daily. Typical news recommender systems in Japan include Google News, Yahoo! News, Smart News, and Gunosy. Users frequently access the news recommender systems to obtain the latest news, which is specific compared with other recommendation domains. For example, Smart News users spend about 16 minutes a day using the service, and 30% of all users launch the service more than 51 times a month [1]. Even before the spread of news recommender systems, news distribution had long been conducted by media such as newspapers and television. While newspapers and television are limited in the amount of news they can distribute, news distribution in online news recommender systems is unlimited in the amount of news they can distribute. In addition, news in newspapers and television distributes the same news to all viewers, while news recommender systems can distribute different news to each user. Thus, compared to older media, online news recommender systems have a higher degree of freedom in the news they deliver. In order to recommend helpful articles to users from among the vast number of news articles that accumulate daily, it is necessary to overcome the challenges of news recommender systems, which arise from the characteristics of news recommendations.

1.2 Characteristics of News Recommender Systems

News recommender systems have unique and common characteristics compared with other domains. Table 1.1 compares the characteristics in the news and other domains from three dimensions: *recency-sensitiveness*, velocity, and sponsored ranking, which are described as follows:

i. Recency-sensitiveness

Recency-sensitiveness is a unique characteristic of news recommender systems. The value of news articles decreases rapidly over time. For example, breaking news of a disaster is considered more valuable if reported immediately than if it is reported a year later. Thus, the recent item has more importance than the older one, and we call this characteristic recency-sensitiveness. Newer items are also considered more valuable to consumers in domains such as movies and music, but

Table 1.1: Characteristics of the recommendation compared with other domains.

	i. Recency-sensitiveness	ii. Velocity	iii. Sponsored Ranking
News	High	High	✓
Book	Low	High	
Clothing	Middle	High	
Music	Middle	High	✓
Movie	Middle	Low	✓
Job	Middle	High	
Hotel	Low	Low	
Web doc.	Low	High	✓

the rate of decreasing value is likely to be slower than for breaking news. Due to the recency-sensitiveness, news recommender systems are required to deliver news articles to users immediately as they become available.

ii. Velocity

News recommender systems have high velocity in that items appear one after another at a high pace. According to Gadepally's definition [44], the velocity in big data is a concept that deals with the speed of the data. In this thesis, we define high-velocity as if the number of new items is more than $10,000$ in a day. In contrast to the news domain, the domain of movie or hotel recommendations has low velocity because the number of new items is less than $10,000$ in a day.

iii. Sponsored Ranking

Many news recommender systems on the web generate revenue by serving ads. Recommender systems of video and music also generate revenue from ads where ads are inserted before the content. They have characteristics that mix organic items and ads and present them to users as rankings, and we call this ranking a *sponsored ranking* that includes ads sponsored by companies. In this ranking that mixes advertisements, it is important to increase user satisfaction by delivering items that are attractive to users and, at the same time, to ensure advertising revenue.

Table 1.2: Challenges of news recommendation compared with other domains.

	i. Timeliness Req.	ii. Cold Start Prob.	iii. Post-click Data Volume	iv. Ad-aware Rec.
News	Highly required	Critical	Low	Required
Book		Critical	High	
Clothing	Required	Critical	Middle	
Music	Required	Critical	Middle	Required
Movie	Required		Middle	Required
Job	Required	Critical	Middle	
Hotel			High	
Web Doc.		Critical	High	Required

1.3 Challenges of News Recommender Systems

Table 1.2 shows the challenges arising from the characteristics of news recommendations. The challenges of news recommender systems are as follows.

i. Timeliness Requirement

Recency-sensitiveness requires timeliness in news recommender systems. In this thesis, we define the timeliness requirement as the requirement that new items should be delivered immediately to users as they become available. In order to build a news recommender system that satisfies the timeliness requirement, the similarity between news articles and users must be calculated in real-time. In addition to responding in real-time, the calculation of similarity between a news article and a user must be done with high accuracy to make a beneficial recommendation. In recent years, embedding-based recommender systems can achieve high accuracy by using machine learning models to convert news articles and users into vectors and calculate the similarity. The key to real-time response using embedding-based news recommender systems is the dimension of embedding. In general, while larger dimensionality can represent news articles and user characteristics with high accuracy, larger dimensionality also increases the computational load, making it challenging to respond in real-time. Therefore, it is crucial to represent news articles and users using small dimensions of embedding for accurate recommendations with timeliness requirements.

ii. Cold Start Problem

In news recommender systems, news articles are accumulated from time to time with a high velocity. Therefore, it is challenging to characterize news articles using user behavior data for new news because there is little user behavior data. This problem is referred to as the cold start problem [72]. In order to deal with this cold start problem, the model of the recommender system must utilize user behavior efficiently. The cold start problem in the news recommendation system should be solved simultaneously with satisfying the timeliness requirement.

iii. Low Volume of Post-click Data

News recommender systems need to deal with a low volume of post-click data. Post-click data refers to the user's behavior after clicking on a recommended item. For example, post-click data in a news recommender system includes article reading behavior, which is the behavior after clicking on an article. This post-click data indirectly expresses the user's satisfaction and is important in evaluating the recommender system. While post-click data is valuable and indirectly expresses user satisfaction, the amount of available data is limited because it is only available after the user clicks on an item. Furthermore, since news articles in news recommender systems change from time to time with a high velocity, the post-click data for a single news article is limited. Therefore, when using post-click data for evaluation in a news recommender system, it is necessary to utilize a small amount of post-click data efficiently.

iv. Ad-aware Recommendation

In existing news recommender systems, news recommendation algorithms select news, and ad recommendation algorithms select ads separately. The news recommender system then delivers the news and advertisements obtained by each separate calculation process to the user as a single ranking. The problem with this approach is that because articles and ads are selected as separate entities, if the selection of articles affects the effectiveness of advertisements, the final ranking might fail to satisfy users and generate revenue optimally. However, it is currently unclear how news articles affect ad consumption in news recommender systems.

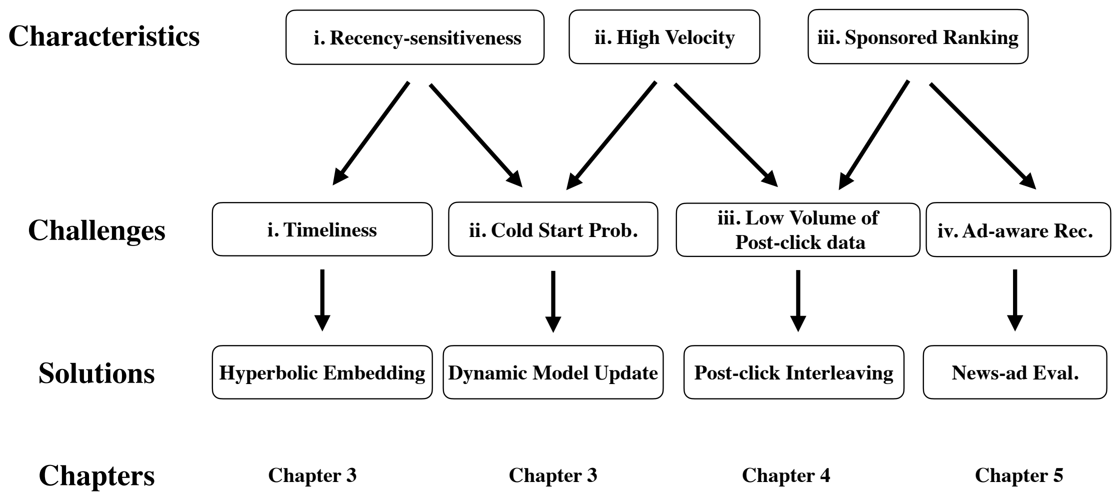

Figure 1.1: Characteristics, challenges, and structure of the thesis.

1.4 Thesis Organization

This thesis aims to address the challenges in news recommender systems. Figure 1.1 shows the relationship between the characteristics, challenges, and structure of the following thesis.

Chapter 2: Related Work

In Chapter 2, we survey works related to the research presented in this thesis. This chapter includes topics of hyperbolic embeddings, online evaluation methods, and sponsored rankings.

Chapter 3: An Accurate and Efficient Embedding Method for News Recommendations

Regarding the challenge of the timeliness requirement, Chapter 3 aims to develop an embedding method that provides highly accurate recommendations while keeping the dimension of the embedding vector as small as possible. Simply reducing the dimensionality of embedding vectors will degrade recommendation accuracy. Therefore, we propose a method to embed recommendation targets into vectors in hyperbolic space, which can accurately capture the similarity structure between users and items even with a small dimension. Furthermore, we aim to improve the accuracy by introducing a regularization term in the learning that utilizes the distribution of user behavior data and the properties of hyperbolic space.

Chapter 3 also proposes a method to update the model of the recommender system in order to reflect user behavior data in the model efficiently regarding the cold start problem. The idea of the method is not to retrain the recommendation model from scratch but to successively update the trained model using newly available user behavior data. By introducing this method, the training can start with the inherited features of the previously trained model, and higher accuracy can be achieved with the same amount of training compared to retrain the model from scratch. In addition, by taking advantage of the fact that the latest news articles are more valuable than those in the past, we aim to reduce the training cost of the model by introducing a process to delete articles older than a certain period from the model.

Chapter 4: An Interleaving Method for Evaluating Post-click Metrics

Regarding the challenge of the low volume of post-click data, Chapter 4 proposes a method for efficiently evaluating recommendation algorithms based on the post-click behavior of a small number of users. While A/B testing is the *de facto* standard method to evaluate post-click behavior, there are cases where evaluation by A/B testing is not easy in news recommender systems. For example, when a large amount of user behavior data on an item is not available, it becomes difficult to evaluate the recommendation algorithm using those items. Therefore, the proposed method aims to reduce variance for efficient evaluation by using importance sampling, in which items with a small number of clicks are preferentially presented to users. This operation achieves a smaller error in the evaluation. Our method can be used in news recommender systems to evaluate the algorithm of news articles and ads.

Chapter 5: Evaluating the Effects of News Articles on Ad Consumption

Regarding the ad-aware recommendation, Chapter 5 evaluates the effect of news articles on ads in news recommender systems. Specifically, we focus on the quality of articles and evaluate the extent to which these impact the effectiveness of advertising. The evaluation will include an offline log-based analysis utilizing user behavior logs and an online evaluation to measure changes in ad consumption when users are provided with different article qualities. The findings from this thesis are helpful to construct ad-aware news recommender systems.

Finally, Chapter 6 summarizes this thesis and shows some interesting directions to be explored in future work.

Chapter 2

Related Work

This chapter provides reviews related to the challenges for the news recommender systems. Section 2.1 reviews hyperbolic embeddings and model update strategies and Section 2.2 introduces efficient online evaluation methods. Finally, Section 2.3 shows the related works about sponsored rankings.

2.1 Embedding-based Recommendation

First, we introduce hyperbolic embeddings. Next, we review model update techniques for real-world recommender systems.

2.1.1 Small Dimension Embeddings

Online news recommender systems must deliver news articles immediately to users as they are available. In order to build a system that satisfies this timeliness requirement, the similarity between news articles and users must be calculated in real-time. Recently, embedding-based recommender systems have been de-facto standards that can achieve high accuracy by using machine learning models to convert items and users into vectors. The key to real-time response using the embedding-based news recommender systems is a small embedding dimension. Hyperbolic embedding has recently been used to model complex networks, which can capture the relationship between users and items even with a small embedding dimension. Nickel and Kiela applied hyperbolic distance to model taxonomic entities and graph nodes [57]. Following this work, various applications and generalization using hyperbolic embeddings have been proposed in the literature. In the information retrieval field, Tay et al. proposed a question-answering system using a hyperbolic space [78]. Hyperbolic models have been also used for recommendations [13, 29, 79]. Shimizu et al. generalize the fundamental components of neural networks in a single hyperbolic geometry model [76]. Nonetheless, there were no investigations into why hyperbolic models work well for recommendation datasets. Thus, in this work, we investigate whether recommendation datasets have a hierarchical or tree-like structure suitable for hyperbolic embeddings.

To improve recommender effectiveness, regularization techniques are frequently used. Matrix factorization (MF) models have many parameters, and to optimize them, a regularization term is added to prevent overfitting. The user and item factors learned through matrix factorization are usually regularized with the L2-norm. Liang et al. used the item–item co-occurrence matrix for regularization [48]. Rendle and Schmidt-Thieme used regularization to learn kernel matrix factorization models [69]. However, very few classical regularization methods are directly applicable or easily generalizable in the Poincaré ball model of hyperbolic space. For instance, the L2 regularization constraint degrades recommendation accuracy because it requires that all nodes remain close to the origin and breaks a hierarchical or tree-like structure [47]. In this paper, we use a graph property—specifically, graph centrality–to maintain hierarchical or tree-like structures for regularization to improve recommender accuracy.

2.1.2 Efficient Model Updating

Efficiency is one challenge for real-world recommender systems. An essential element of a practical recommender system is handling the dynamic nature of incoming data, for which timeliness is crucial consideration. Because it is prohibitive to retrain the full model online, various studies have developed incremental learning strategies for neighbor-based, graph-based, and probabilistic MF methods [36]. In this study, we propose dynamic model updates for hyperbolic embeddings rather than batch updates inspired by previous incremental learning strategies and aim to improve efficiency and robustness. We note that previous hyperbolic models [13, 29, 79] are difficult to apply to the real-time stream recommender system because of their computational resource limitations. Thus, we focused on building a hyperbolic model for efficient recommendations with dynamic updates.

2.2 Post-click Evaluation

Online evaluations are the basis of data-driven decision-making [26]. A typical online evaluation method is A/B testing [46], while there are also alternatives such as interleaving and bandit algorithms.

2.2.1 Interleaving Methods

Interleaving (or *multileaving*, which refers to interleaving of more than two rankings) is a method used to increase the efficiency of online evaluation. Interleaving was reported to be 10 to 100 times more efficient than A/B testing [67, 15]. Schuth et al. proposed two multileaving methods called team draft multileaving (TDM) and optimized multileaving (OM) [75]. Recently, sample-scored-only multileaving (SOSM) [10] and pairwise preference multileaving (PPM) [61] were developed as more scalable multileaving methods. While PPM designed to evaluate click-based metrics is the state-of-the-art method in the interleaving, it is not trivial how to modify PPM to evaluate post-click metrics. Specifically, Schuth et al. found that a simple extension of the TDM resulted in a low accuracy in the non-click-based metrics (e.g., time to click metrics) [74]. Our approach differs from previous interleaving methods because we formulated a new *credit function*, which is used to aggregate user feedback in the interleaving, as the expectation of post-click metrics for accurate evaluation.

2.2.2 Variance Reduction

Variance reduction is a technique commonly used to improve efficiency in online evaluations. Poyarkov et al. [64] used a boosted decision tree regression to reduce the variance by matching similar users. Deng and Shi [26] improved evaluation efficiency by utilizing pre-experiment data through variance reduction. Xie and Aurisset implemented stratification for variance reduction [87]. These studies reduced the variance for each tested group in a bucket-level evaluation like A/B testing. In contrast to bucket-level variance reduction, we reduce the item-level variances in post-click values based on the variance of the post-click values differing greatly for each item. By reducing item-level variances, we were able to improve the evaluation efficiency. Oosterhuis and de Rijke applied variance reduction to an efficient evaluation of CTR [62]. However, it was unclear how to apply this method to a post-click metric, which we focus on in this work. Oosterhuis's method [62] relies heavily on the examination probability and has only been validated by simulation-based experiments. We found that the evaluation accuracy of the post-click metrics was highly compromised when there was an estimation error in the examination probability of the real service data. We developed a stabilization technique to address this problem. The stabilization technique reduces the systematic error caused by the estimation error of the examination probability.

2.2.3 Off-policy Evaluation

Recently, several off-policy evaluation methods have been developed [33, 43, 84]. The goal of off-policy evaluation is to estimate the performance of a policy from the data generated by another policies in reinforcement learning [28]. Specifically, Saito proposed an unbiased estimator for post-click metrics [71]. Our method is different in that we interleave rankings dynamically so that the estimation error is minimized by the variance reduction in an on-policy manner.

2.2.4 Bandit Algorithms

One common form of *exploration* vs. *exploitation* trade-off in the reinforcement learning fields is called the *multiarmed bandit problem* where the world can be imagined as a collection of slot machines, each with a different but unknown payoff, and the player must play, repeatedly, to maximize his wealth according to the Burtini's survey [11]. Algorithms used for bandit problems are similar to the proposed method of evaluating multiple rankings in this study. However, the objective of a bandit algorithm is different. The objective of a bandit algorithm is to maximize the total gain during a specific period or to identify the best arm. The objective of our study is to evaluate the superiority or inferiority of each pair of rankings. The evaluation result for each pair cannot be obtained from a bandit algorithm. However, the evaluation result for each pair may help decision-makers choose the best ranking in terms of effectiveness and the cost of producing the rankings.

2.3 Sponsored Ranking Evaluation

This section reviews studies on two research topics related to the analysis of news article and ad consumption: user behavior analysis and optimization of organic content and ads.

2.3.1 User Behavior Analysis

In the field of information retrieval and recommendation systems, users' click behaviors have been studied for a long time [27, 38, 63]. These studies were conducted from different view points, including content position [41, 42], reliability [90], quality [90], presentation [82], and the delivery mechanism of the system [49]. In this study, we investigate users' click behaviors and ad consumption from the viewpoint of the quality of news articles.

There are reports that the quality of contents in the web site (e.g., news articles and ads) affects user behaviors. Lu et al. showed that low-quality articles are more likely to be clicked [50]. Lu et al. also investigated the effects of negative user experience in mobile news services and showed that negative experiences reduced user satisfaction [51]. Mao et al. showed that the usefulness of search results correlates with user satisfaction [53]. Alanazi et al. investigated the impact of ad quality on mobile search engine result pages (SERPs) and showed that users pay different amounts of attention to organic results and SERPs for different levels of ad quality [3]. Cramer reported that high-quality native ads could still have a negative effect on perceived site quality if they were too content-relevant [22].

While the effect of the content quality on content own consumption has been investigated, the effect of news quality on ad consumption has not yet been studied in the literature, and is first revealed in this work.

2.3.2 Optimization of Organic Content and Advertisements

Web services, such as social network services and news services, provide users with organic content and advertisements. Organic content attracts users, while advertisements are a source of revenue for many companies. Both are vital for sustaining services. Therefore, there are many studies focusing on the optimization of user satisfaction from organic content and that of revenue from advertising. Zhao et al. proposed a method to generate a ranking that considers long-term user experience from organic content and profits from advertisements [91]. Wang et al. proposed a reinforcement-learning-based method that dynamically inserts advertisements rather than fixing the position and number of advertisements to be inserted [83]. Yan et al. proposed a method for optimizing the location of ads inserted between articles in a news feed [88].

These studies treated organic content and advertisements as independent from each other when applied to real-world services. In this study, we show that the quality of news articles affects ad consumption. If we built our insight into the optimization problem, the optimization would be performed more effectively.

Chapter 3

An Accurate and Efficient Embedding Method for News Recommendations

3.1 Introduction

Although recent studies show that embedding-based recommender algorithms have achieved high levels of accuracy, how to learn embeddings in real-world recommender settings is one of the remaining challenges [59, 32]. In particular, there are special requirements for real-time stream recommender systems with a large number of new items added continuously in short intervals, such as news applications and social network services: 1) compact embeddings for avoiding performance degradation in response time [40, 59]; 2) online learning algorithms that can incrementally update the embeddings [36]; and 3) sufficiently high recommendation accuracy to allow users to find what they need within a large volume of information [39].

To satisfy the above requirements for a real-time stream recommender system, *hyperbolic embeddings* are considered as a reasonable solution, since they have recently been reported to be compact and able to produce highly accurate embeddings [57]. Hyperbolic embeddings are known to perform well when learning with hierarchical or tree-like structure data, even in low-dimensional spaces. However, there remain several questions in applying hyperbolic embeddings to a real recommender service: 1) whether the existing recommendation datasets have a suitable hierarchical or tree-like structure to embed an item in a low-dimensional space; 2) how online learning should be conducted without sacrificing the recommendation accuracy; and 3) whether hyperbolic embeddings actually lead to a better recommendation accuracy and user satisfaction when they are applied to a real recommendation service.

In this study, we first investigated whether the graph of recommendation datasets has a hierarchical or tree-like structure. While they were shown to form such a structure, we also found simple application of the hyperbolic embeddings did not allow us to learn embeddings that adequately reflected the hierarchical structure. Therefore, we introduced a regularization technique to promote embeddings that reflects a hierarchical structure using graph centrality in the loss function. Next, we proposed a model update technique for online hyperbolic learning. In this method, we only update embeddings of the most recently added items while keeping the previously added embeddings unchanged.

We conducted an offline experiment and A/B testing to evaluate the two proposed techniques. Throughout the offline experiment, we used both public datasets and a proprietary dataset generated by a million-scale news service. In the offline evaluation, we confirmed that regularization that leveraged the graph centrality improved the ranking metrics. Furthermore, we simulated real-time stream recommender systems and found that the dynamic model update improved the learning efficiency and robustness. Finally, we evaluated the performance of our method in a real-world news application by conducting A/B testing. The results showed that the number of clicks rose significantly (+3%), and dwell time also rose (+10%).

Our Contributions. The key contributions of our work are summarized as follows:

- **Analysis**: We investigated the existence of a hierarchical and tree-like structure in the graphs of various recommendation datasets. All graphs of the recommendation datasets that we investigated showed high hierarchical or tree-like structure metrics.

- **Accuracy**: We proposed regularization using graph centrality for hyperbolic embeddings. By introducing this regularization, we improved the performance of recommender systems.

- **Efficiency**: We proposed a dynamic model update strategy for real-time stream recommender systems. The empirical simulation results showed that this method achieved not only greater efficiency but also greater robustness.
- **Application**: To the best of our knowledge, this is the first work to deploy a hyperbolic embedding method in a real-time stream recommender system. We deployed our method to a million-scale news recommendation service and confirmed that the number of clicks and user dwell time increased significantly.

The remainder of this chapter is organized as follows. In Section 3.2, we report the results of our analysis of hierarchical or tree-like structure in the recommendation datasets. We propose a regularization technique in Section 3.3 and model update strategy in Section 3.4. In Sections 3.5 and 3.6, we present the results of the offline experiment and A/B testing, respectively. In Section 3.7, we conclude our work and suggest directions of future work.

3.2 Analysis

In this section, we investigate whether recommendation datasets have a hierarchical or tree-like structure.

3.2.1 Notation

To represent the user and item interactions in recommender datasets, we introduce a directed graph $G = (V, E)$, where V is a set of nodes that represents all of the items, and E is a set of edges that individually connect one item to another. $|V|$ and $|E|$ represent numbers of nodes and edges, respectively. In this work, we constructed edges based on users' interactions and connected items if they were adjacent in a user's history of interactions. More formally, letting $\mathbf{i} = (i_1, i_2, \ldots)$ be a sequence of items a user has interacted with, sorted by the interaction time, $E = \bigcup_{\mathbf{i}} \{(i_k, i_{k+1}) | k = 1, 2, \ldots\}$. We use this graph definition throughout this paper.

3.2.2 Datasets

In this study, we took datasets from diverse domains, including movie, news, and e-commerce services, with details as follows:

- **MovieLens20M**: This is a widely used dataset from GroupLens Research for evaluating recommender algorithms. This dataset contains ratings of movies[1]. For each user, item IDs with a rating more than 1.2 times the average value were extracted as positive feedback.
- **Online Retail** [16]: This is an e-commerce dataset that contains all the transactions occurring between Januaray 12, 2010 and September 12, 2011 for a UK-based and registered non-store online retail site[2]. The company primarily sells unique all-occasion gifts. Many customers of the company are wholesalers.
- **Amazon Reviews** [56]: This dataset contains product reviews from Amazon. We selected five categories, *Sports*, *Cell Phones*, *Games*, *Automotive*, and *Clothing*, by considering the diversity of domains and dataset size[3].
- **Lucra**: This is a news application service dataset. It is one of the most popular news application services among female users in a country served by Gunosy. The news application dataset is different from the other datasets in that the recommendation items are frequently changed.

We filtered out users who had interactions with items less than three times and only used each user's last ten interactions in the analysis. Table 3.1 shows the statistics of the datasets.

3.2.3 Metrics

Here, we introduce the graph metrics related to hierarchical or tree-like structures.

In hierarchical networks, the degree of clustering can be used as an indicator of a hierarchical structure. *Clustering coefficient* [85] is a measure of the degree to

[1]https://grouplens.org/datasets/movielens/

[2]http://archive.ics.uci.edu/ml/datasets/Online+Retail.

[3]Datasets were obtained from http://jmcauley.ucsd.edu/data/amazon/ using the five-core setting, with the domain names truncated in the interest of space.

Table 3.1: Dataset statistics.

	$\lvert V\rvert$	$\lvert E\rvert$	Degree	Density	C(k)
MovieLens	8,026	83,400	20.7	0.00258	0.0717
Online Retail	2,660	23,810	17.9	0.00673	0.0781
Sports	14,576	100,095	13.7	0.000942	0.0412
Cell Phones	26,386	169,920	12.9	0.000488	0.0364
Games	25,026	157,167	12.6	0.000501	0.0303
Automotive	24,771	167,236	13.5	0.000545	0.0334
Clothing	8,597	37,748	8.78	0.00102	0.0495
Lucra	986	23,282	47.2	0.0479	0.491

which nodes in a graph tend to cluster together and is defined as follows:

$$C(k_i) = \frac{1}{|V|}\sum_{i=1}^{|V|} \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i\,(k_i - 1)}, \tag{3.1}$$

where N_i is a set of nodes that are connected to v_i, and $k_i = |N_i|$.

Gromov's hyperbolicity δ was used for the analysis of tree-like structures [2]. The smaller δ is, the closer the graph is to a tree, and it becomes a tree when $\delta = 0$ [14]. More formally, Gromov's hyperbolicity δ is defined as follows [2]. Let u, v, w, and x be nodes in the graph G. Consider the three distance sums, $dG(u, v) + dG(w, x)$, $dG(u, w) + dG(v, x)$, and $dG(u, x) + dG(v, w)$. We denote S_1, S_2, and S_3 that are sorted in non-decreasing order of the three distance sums: $S_1 \le S_2 \le S_3$ and denote the hyperbolicity of a quadruplet u, v, w and x as $\delta(u, v, w, x) = (S_3 - S_2)/2$. Then, δ-hyperbolicity in the graph G is the maximum hyperbolicity over all possible quadruplets $u, v, w, x \in V$, or $\delta(G) = \max \delta(u, v, w, x)$. As the computational time of the δ-hyperbolicity is $O(|V|^4)$, it is not easy to obtain the exact value of the δ-hyperbolicity. Thus, we approximately computed the δ-hyperbolicity by sampling 100,000 quadruplet nodes. Note that we ignored the δ value if a selected quadruplet node was not connected following to a previous work [14].

3.2.4 Results

Figure 3.1 shows the degree distributions, which reveal that all of the datasets

Table 3.2: The relative frequency of δ-Hyperbolicity quadruplets.

δ-Hyperbolicity	MovieLens	Online Retail	Sports	Cell Phones	Games	Automotive	Clothing	Lucra
0	6.38e-01	5.92e-01	6.06e-01	6.07e-01	6.09e-01	6.22e-01	6.19e-01	6.79e-01
0.5	3.55e-01	3.92e-01	3.81e-01	3.80e-01	3.79e-01	3.68e-01	3.70e-01	3.18e-01
1	6.58e-03	1.46e-02	1.18e-02	1.11e-02	1.14e-02	8.74e-03	1.04e-02	1.82e-03
1.5	-	1.00e-05	2.18e-05	2.01e-05	1.02e-05	1.02e-05	3.29e-05	-
max δ	1	1.5	1.5	1.5	1.5	1.5	1.5	1

had a power law distribution. A network with a degree distribution that follows a power law is called a scale-free network [6].

Figure 3.2 shows the clustering coefficient plots. When a graph has a scale-free property, it has been empirically shown that $C(k) \simeq k^{-\beta}$ if the graph is hierarchical [68, 18, 45]. In Figure 3.2, most of our datasets had linear fittings in which the coefficient of determination was negative (excluding MovieLens and Lucra), and thus, the graphs had hierarchical structures.

Table 3.2 shows the relative frequency of δ-hyperbolicity values from sampled nodes, which were low for all of the datasets. More than 98% of the quadruplets had $\delta \leq 1$, indicating that all of the graphs were metrically close to trees. In particular, MovieLens and Lucra had δ-hyperbolicity of 1, meaning that they had a chordal graph structure. The other two datasets had δ-hyperbolicity of 1.5. Because k-chordal graphs have hyperbolicity at most $k/4$, the other two datasets were 6-chordal graphs, which indicates the graph did not contain any chordless 7-cycles [86]. Thus, all the datasets were metrically close to trees.

In summary, the examined recommendation datasets were found to have hierarchical or tree-like structures in the graph suitable for hyperbolic embeddings.

3.3 Regularization

In this section, we propose a regularization technique for hyperbolic embedding using graph-central structures.

3.3.1 Hyperbolic Embeddings

The Poincaré ball model is the Riemannian manifold $\mathcal{P}^d = (\mathcal{B}^d, g_x)$, in which $\mathcal{B}^d = \{\mathbf{x} \in \mathbb{R}^d : ||x|| < 1\}$ is the open d-dimensional unit ball that is equipped

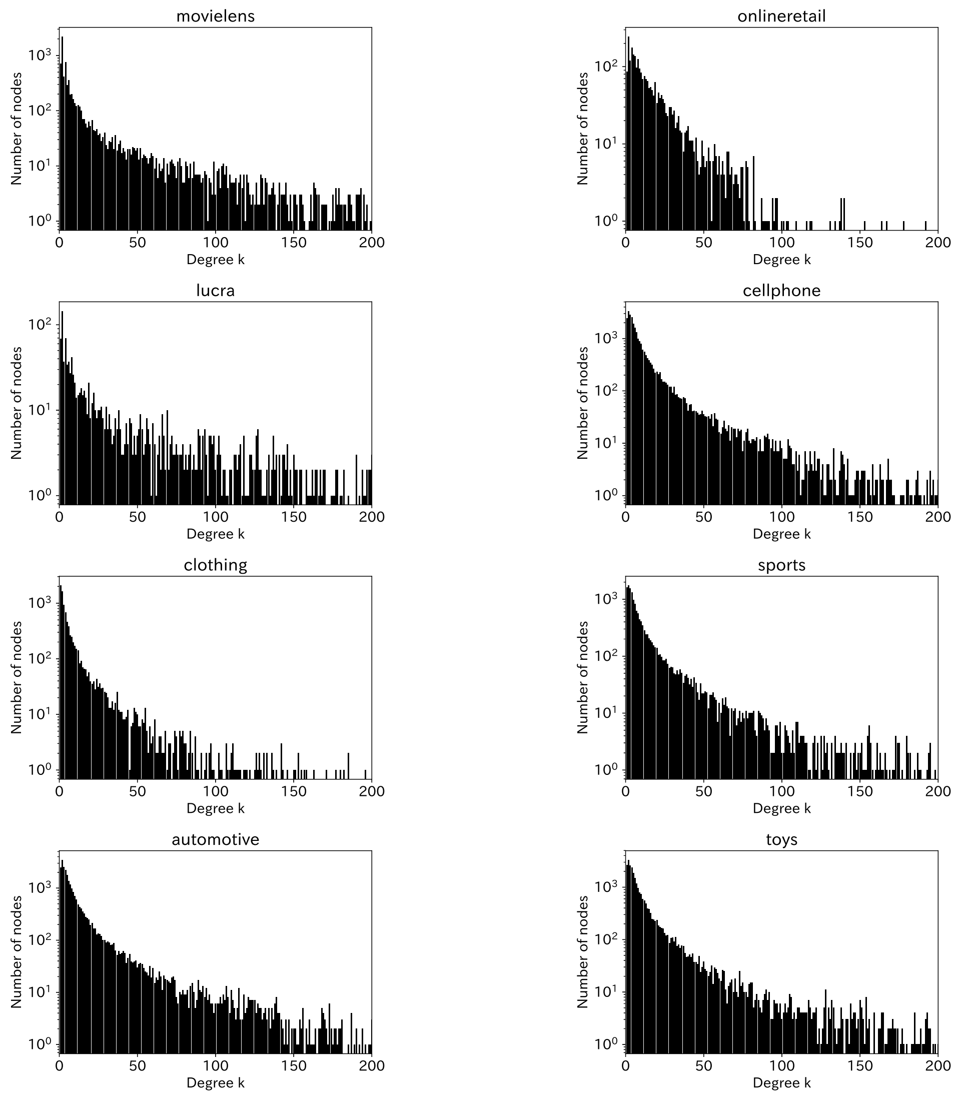

Figure 3.1: The degree distributions.

with the following metric:

$$g_{\mathbf{x}} = \left(\frac{2}{1 - ||\mathbf{x}||}\right)^2 g^E, \tag{3.2}$$

where $\mathbf{x} \in \mathcal{B}^d$ and g^E denotes the Euclidean metric tensor. The distance between points $\mathbf{v}$, $\mathbf{u} \in \mathcal{B}^d$ is given by

$$d(\mathbf{u}, \mathbf{v}) = \operatorname{arcosh}\left(1 + 2\frac{||\mathbf{u} - \mathbf{v}||^2}{(1 - ||\mathbf{u}||^2)(1 - ||\mathbf{v}||^2)}\right). \tag{3.3}$$

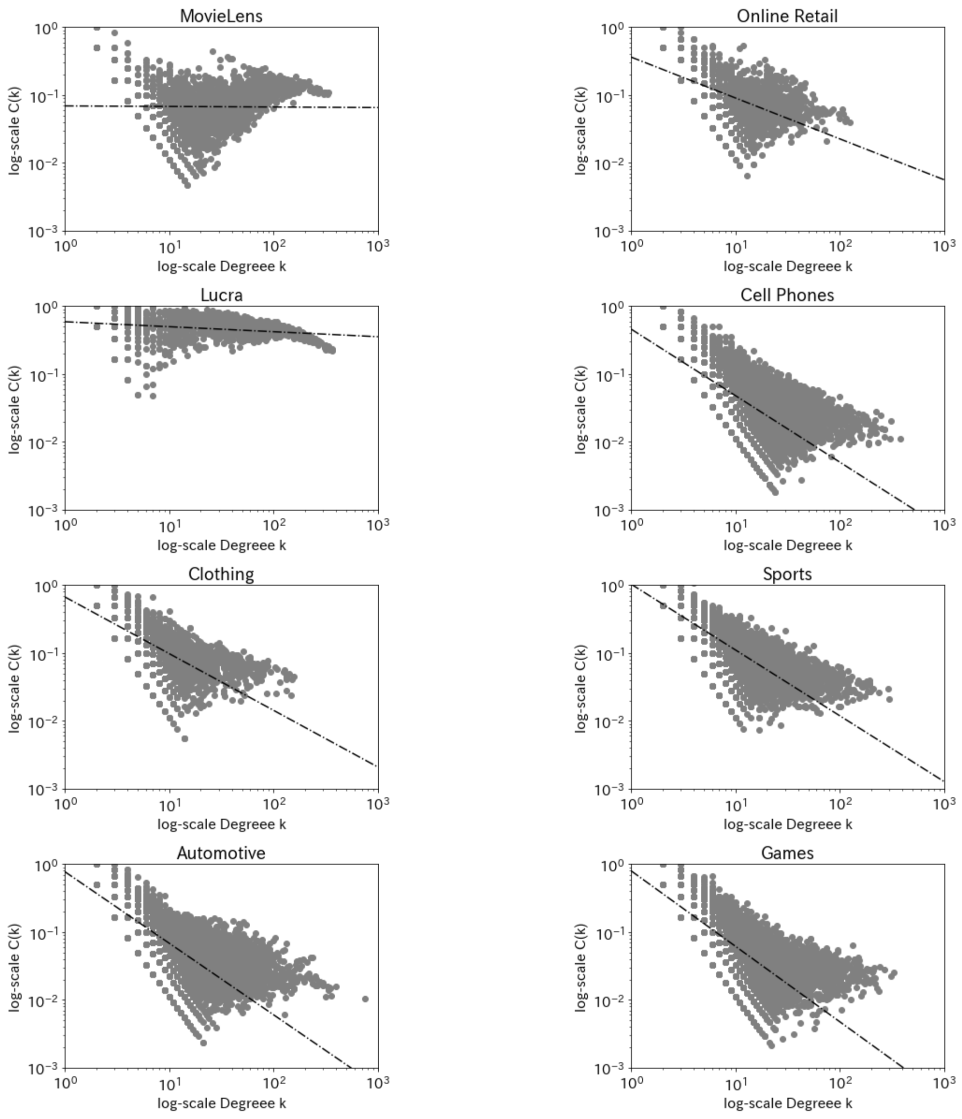

Figure 3.2: The clustering coefficient plots.

Poincaré embedding determines the embedding $\Theta = \{\theta_i\}_{i=1}^{n} \in \mathbb{R}^{n \times d}$, where n is the number of all the nodes and d is the embedding size by solving the following optimization problem:

$$\arg\min_{\Theta} \mathcal{L}(\Theta) \text{ s.t. } \forall \theta_i \in \Theta : ||\theta_i|| < 1. \tag{3.4}$$

Here, θ_i corresponds to each node in V. We use the same loss function as Ref. [57], which uses the negative samples $\mathcal{N}(u) \subset \{v : (u, v) \notin E, v \neq u\}$ to maximize the

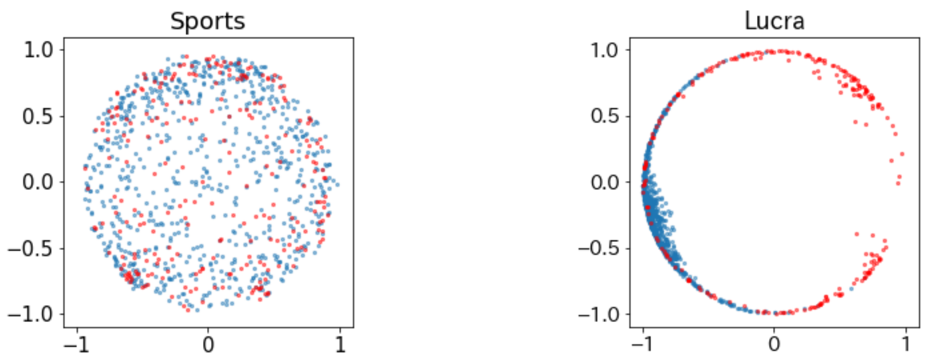

Figure 3.3: The 2D item embedding visualization with and without regularization. The top 25% high-degree nodes are plotted as red dots and the remainder as blue dots.

distance between the embeddings of unrelated objects:

$$\mathcal{L}(\Theta) = -\sum_{(u,v)\in E} \log \frac{e^{-d(u,v)}}{\sum_{v'\in\mathcal{N}(u)} e^{-d(u,v')}}. \tag{3.5}$$

Hyperbolic embeddings work for the graphs of hierarchical or tree structure because of the following two properties: 1) Hyperbolic space expands as one moves from the center of the sphere to the boundary. 2) The number of nodes at the top of the hierarchical structure or near the tree's root is fewer than the nodes at the bottom of the hierarchical structure or the tree leaves. Thus, nodes higher in the hierarchy or closer to the tree's root are expected to be embedded near the center of the sphere, while nodes lower in the hierarchy or closer to the leaves of the tree are expected to be embedded near the boundary.

3.3.2 Regularization

As we confirmed that the graphs of recommendation datasets had hierarchical or tree-like structures in Section 3.2, we investigated how each node was embedded by the hyperbolic embeddings method. It can be seen that nodes with high graph-centrality are embedded near the boundary as well as those with a low degree in Figure 3.3.

Inspired by the related work [58], we hypothesize that a node that is similar to many other nodes should be embedded near the center of the ball. In this recommendation dataset, a node that is similar to many other nodes is the same as the node degree is large. The graph in the recommendation dataset is constructed by connecting the nodes interacted by a user, i.e., nodes are connected according to the similarity of the user's interests. However, without the regularization, the large degree nodes are embedded near the boundary in Figure 3.3.

Thus, we introduce the L2-based regularization term $\mathcal{L}_R$ to embed the large degree nodes near the center of the ball:

$$\mathcal{L}_R(\Theta) = \sum_{u \in V} w_u ||\theta_u||^2, \tag{3.6}$$

where w_u represents the graph-centrality metric of node u. We used the L2-based regularization because the embedding accuracy might be decayed if we use the geometrically defined regularization, such as the distance in the hyperbolic space. Due to the structure of the Poincaré ball model, the distance of points increases exponentially the closer they are to the boundary. The ideal embedding in hyperbolic space is a situation where the nodes at the top of the hierarchical structure are placed at the center of the space, and the nodes at the bottom of the hierarchical structure are placed near the boundary away from the origin [57]. When we use the geometrically defined regularization, all nodes might be closer to the origin rather than the boundary because the penalty of nodes at the boundary is too high. Then, the hierarchical structure can not be fully reflected that the recommendation dataset has. We note that the standard L2 regularization is not also enough to improve embedding accuracy because all nodes are enforced to be close to the origin and can impair the benefit of hierarchical structures. On the contrary, the weighted L2 regularization term increases the penalty for distance from the center of the sphere for nodes with higher graph centrality. In other words, nodes with a high graph-centrality move closer to the origin, and nodes with a low graph-centrality move closer to the boundary.

The overall loss function is $\hat{\mathcal{L}}(\Theta) = \mathcal{L}(\Theta) + h\mathcal{L}_R(\Theta)$, where h is a hyper-parameter for regularization.

Although there are multiple graph-centrality metrics, such as closeness, degree, and eigenvector centrality, we used the degree centrality. Because real-world recommender systems have large graphs, the computational cost for calculating graph centrality needs to be low. Another reason is the following. The ideal embedding in hyperbolic space is that the highly general nodes at the top of the hierarchy are placed near the origin, while the less general nodes at the bottom of the hierarchy are placed near the boundary [58]. For example, in the case of a graph embedding for the sending relationship of employees' e-mails, the engineering manager is embedded near the origin, and the staff engineers are placed near the boundary. In this paper, we approximate the generality in the recommendation dataset by the node degree. For example, it is natural to think that a movie with a large degree that many users watched (e.x., Titanic) has a high generality, while a niche movie with a small degree has a low generality.

Additionally, by using the degree for weight $w = w_{in} + w_{out}$ (with w_{in} denoting the in degree and w_{out} the out degree), a single embedding update for each edge $(u, v) \in E$ can be calculated without w_u by using the following relationship:

$$\begin{aligned}\mathcal{L}_R(\Theta) &= \sum_{u \in V} w_u ||\theta_u||^2 \\ &= \sum_{u \in V} (w_{u,in} + w_{u,out}) ||\theta_u||^2 \\ &= \sum_{u \in V} (|\{i : (i,u) \in E\}| + |\{i : (u,i) \in E\}|) ||\theta_u||^2 \\ &= \sum_{(u,v) \in E} (||\theta_u||^2 + ||\theta_v||^2)\end{aligned} \tag{3.7}$$

The transformation from the third to the fourth line of the formula is calculated by rounding out the common edges that appear in the first and second terms.

3.4 Dynamic Embedding

In this section, we introduce a dynamic model update technique to apply the hyperbolic embedding method to real-time stream recommender systems.

3.4.1 Problem Definition

In this problem setting, new items and their relationships are provided at every time step. Letting $G_t = (V_t, E_t)$ represent the items and relations that appear at time t, our problem is to learn the node embedding $\Theta_t \in \mathbb{R}^{n \times d}$ based on the data available up to time t. Our goal is to find an efficient and robust solution to this problem.

A straightforward approach for this problem is to learn the embedding $\Omega_t \in \mathbb{R}^{n \times d}$ by using the complete set of data up to time t (i.e., $G_1, G_2, \cdots, G_t$). We call this approach the *oracle*. This approach can be effective but could be infeasible for real-time stream recommender systems that regularly update the embeddings.

In order to clarify the goal of this work, we introduce an effectiveness measure for embedding, $f : \mathbb{R}^{n \times d} \to \mathbb{R}$. This function takes the embedding and measures its effectiveness at a specific task. For example, it can be nDCG@10 in a recommendation task. The robustness of the solution can be considered the closeness to the effectiveness of the *oracle* embedding even when the entire set of data is not used for the embedding learning. Hence, the robustness is measured by $1/(1 + f(\Omega_t) - f(\Theta_t))$. In the following subsections, we propose techniques to close the gap between Ω_t and Θ_t and increase robustness.

3.4.2 Dynamic Model Update

We propose a dynamic model update strategy for real-time stream recommender systems. Since new items are added regularly in real-time stream recommender systems, it is necessary to obtain an embedding of new items as soon as they arrive. However, it is infeasible to use all of the data up to time t (i.e., $G_1, G_2, \cdots, G_t$), at every time step (i.e., the *oracle* embedding), but it can be less effective than *oracle* to use only the data obtained at time t (i.e., G_t) for training the embedding of V_t. This *batch learning* approach using only G_t can result in a considerable inconsistency of embeddings between different time steps; for example, similar items that appear at different time steps may be embedded very differently. These problems imply low effectiveness of batch learning relative to *oracle* embedding, and this accounts for its low robustness.

In contrast, our dynamic model update strategy utilizes only the latest data from multiple time steps and keeps updating recently learned embeddings until they become stable. The key idea of our solution is to update only the embeddings of items that appear within window size τ by using items and their relationships within the same window, as well as those necessary to learn the

Algorithm 1: Dynamic model updates

```
  Input: $G_t = (V_t, E_t)$ for $t = 1, 2, \ldots$.
1 Randomly initialize $\Theta_0$
2 $V_0 \longleftarrow \emptyset$, $E_0 \longleftarrow \emptyset$
3 $V_{0:\tau} \longleftarrow \emptyset$, $E_{0:\tau} \longleftarrow \emptyset$
4 for $t = \tau + 1, \tau + 2, \ldots$ do
5     $V_{t-\tau:t} \longleftarrow (V_{t-\tau-1:t-1} \setminus V_{t-\tau-1}) \cup V_t$
6     $E'_t \longleftarrow \{(u, v) \in E_t : u \in V_{t-\tau:t} \vee v \in V_{t-\tau:t}\}$
7     $E_{t-\tau:t} \longleftarrow (E_{t-\tau-1:t-1} \setminus E_{t-\tau-1}) \cup E'_t$
8     $\Theta_t \longleftarrow \Theta_{t-1}$
9     $\Theta_t \longleftarrow$ UpdateEmbedding($G_{t-\tau:t}, \Theta_t$)
  end
```

embeddings for the new items. More precisely, at time t, we only update the embeddings of $V_{t-\tau}, V_{t-\tau+1}, \cdots, V_t$, with the data obtained between time $t - \tau$ and t. In addition, we incorporate existing embeddings of nodes connected by the relationships within the time window. These nodes and edges are formally defined as $G_{t-\tau:t} = (V_{t-\tau:t}, E_{t-\tau:t})$, where $V_{t-\tau:t} = \cup_{t'=t-\tau}^{t} V_{t'}$ and $E_{t-\tau:t}$ is a set of edges that appear within the time window $[t - \tau, t]$ and consist of at least a node from $V_{t-\tau:t}$. In other words, $E_{t-\tau:t} = \{(u, v) \in \cup_{t'=t-\tau}^{t} E_{t'} : u \in V_{t-\tau:t} \vee v \in V_{t-\tau:t}\}$. By these means, we ensure the consistency of embeddings between new items and existing items.

In practice, the embeddings are learned through Algorithm 1. Lines 1–3 initialize the variables used in this algorithm. Lines 5–7 update nodes $V_{t-\tau:t}$ and edges $E_{t-\tau:t}$ to learn the embeddings at time t. Line 8 copies the embedding from the previous time step to that at time t for initialization. Line 9 learns the embeddings with $G_{t-\tau:t} = (V_{t-\tau:t}, E_{t-\tau:t})$ based on the initialized embedding Θ_t. In the embedding learning process, nodes in $V_{t-\tau:t}$ are updated based on their initialized embedding and edges $E_{t-\tau:t}$, while nodes that are not in $V_{t-\tau:t}$ remain unchanged.

3.5 Offline Experiment

In this section, we describe how we conducted experiments to answer the following research questions:

Table 3.3: The recommendation accuracy using nDCG@10 and HR@10. In this table, L2 means the the L2 regularization and GL2 means the graph-centralized L2 regularization. Poincaré-10 means a 10-dimension Poincaré embedding.

	nDCG@10							
	MovieLens	Online Retail	Sports	Cell Phones	Games	Automotive	Clothing	Lucra
Item-KNN	0.427	0.462	0.293	0.290	0.275	0.304	0.658	0.600
Item2vec	0.383	0.337	0.539	0.294	0.335	0.326	0.659	0.674
GRU4REC	0.333	0.271	0.371	0.330	0.350	0.345	0.538	0.627
NCF	0.378	0.364	0.328	0.293	0.309	0.318	0.603	0.759
LightGCN	0.341	0.269	0.533	0.395	$\underline{0.431}$	0.424	$\underline{0.678}$	0.502
Poincaré	0.459	0.481	$\underline{0.587}$	0.418	0.415	0.438	0.674	0.764
Poincaré L2	$\underline{0.467}$	$\underline{0.485}$	$\underline{0.587}$	$\underline{0.422}$	0.416	$\underline{0.442}$	0.675	$\underline{0.765}$
Poincaré-10 GL2	0.462	**0.490**	0.607	0.434	0.425	0.450	0.686	0.776
Poincaré GL2	**0.470**	**0.490**	**0.608**	**0.435**	**0.433**	**0.457**	**0.696**	**0.779**
	HR@10							
	MovieLens	Online Retail	Sports	Cell Phones	Games	Automotive	Clothing	Lucra
Item-KNN	0.652	0.660	0.533	0.527	0.516	0.542	0.776	0.809
Item2vec	0.640	0.639	0.689	0.584	0.591	0.589	0.769	0.888
GRU4REC	0.644	0.535	0.687	0.623	0.660	0.657	0.806	0.937
NCF	0.694	0.658	0.625	0.625	0.575	0.620	0.773	$\underline{0.954}$
LightGCN	0.628	0.537	$\underline{0.773}$	0.676	$\underline{0.685}$	$\underline{0.685}$	$\underline{0.827}$	0.780
Poincaré	0.744	0.721	0.761	0.680	0.639	0.664	0.784	0.943
Poincaré L2	$\underline{0.749}$	$\underline{0.726}$	0.761	$\underline{0.685}$	0.644	0.668	0.786	0.942
Poincaré-10 GL2	0.759	0.726	0.804	**0.734**	0.690	0.710	0.827	0.967
Poincaré GL2	**0.769**	**0.733**	**0.812**	**0.734**	**0.699**	**0.714**	**0.834**	**0.968**

RQ1 Is our regularization effective at improving evaluation accuracy of recommendation?

RQ2 How efficient and robust is our dynamic embedding?

We called the first task answering **RQ1** the *accuracy evaluation* and the second task answer **RQ2** the *efficiency evaluation.*

3.5.1 Metrics

We used the normalized discounted cumulative gain (nDCG@10) and hit ratio (HR@10) as the primary evaluation metrics. Both are well-established ranking metrics for recommendation. We evaluate the methods with user sessions. A session in this context means an event that a user visits a recommendation service and clicks on one of the items in the recommended list. Let S be the set of sessions for the evaluation, $c_{s,i}$ be 1 when the item at position i is clicked, and 0 otherwise.

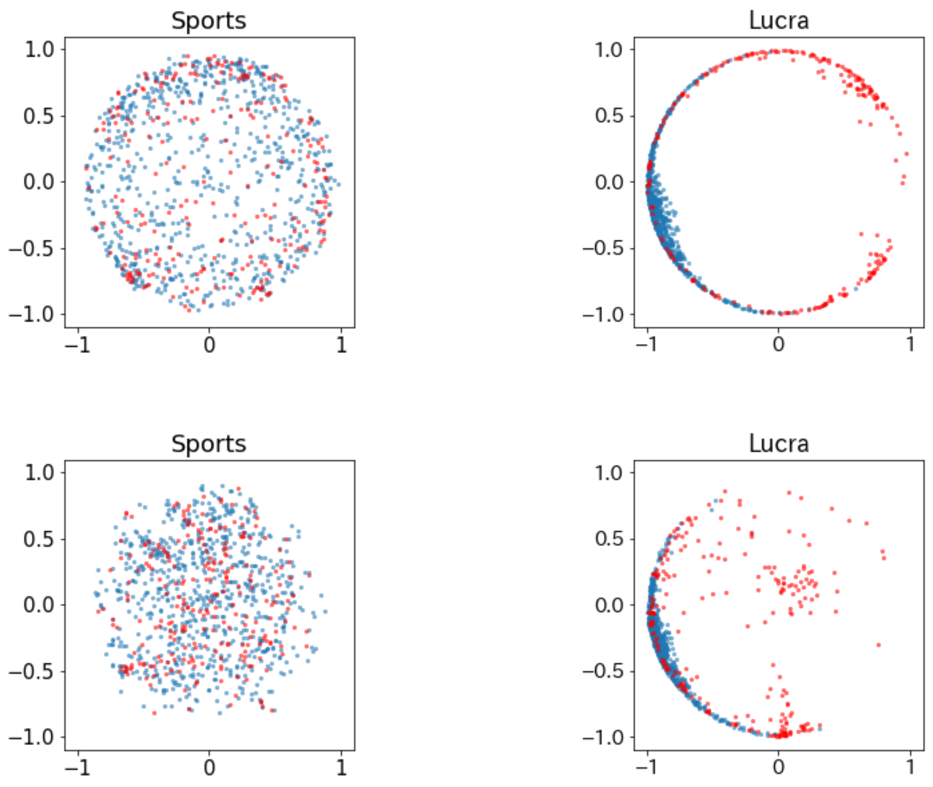

Figure 3.4: The 2D item embedding visualization with and without regularization. The upper side of each dataset shows the results without any regularization. The lower side shows results with graph-centralized regularization.

Each metric is formulated as:

$$\text{nDCG@}k \quad = \frac{1}{|S|}\sum_{s\in S}\frac{\sum_{i=1}^{k} c_{s,i}/\log_2(i+1)}{\max_\pi \sum_{i=1}^{k} c_{s,\pi(i)}/\log_2(\pi(i)+1)} \tag{3.8}$$

$$\text{HR@}k \quad = \frac{1}{|S|}\sum_{s\in S}\min(|\{i|c_{s,i}=1, i\leq k\}|, 1) \tag{3.9}$$

where π is an arbitrary permutation of positions.

We constructed a ranking for a user by selecting a positive sample item that the user clicked on and 20 negative sample items that the user did not click on.

3.5.2 Accuracy Evaluation

Settings

For all the datasets, the test sets were composed of the last item on which the user clicked, and these items were excluded from the train and validation sets for each user. We split the dataset into train : validation : test = 80 : 10 : 10. The models were trained with the training set for 20 epochs and then evaluated with the test set. We used the following common parameter settings: vector dimension (10, 100, and 300), learning rate (0.001, 0.01, 0.1, and 1.0), batch size (128, 256, and 512), and negative sampling size (5, 10, 15, and 20). Ablation study w.r.t. hyper-parameters were tuned with the validation set.

We compared the results using models as follows:

- **Item-KNN** [9]: A traditional collaborative filtering (CF) approach based on k-nearest-neighbor (KNN) and item–item similarities. Regularization was also included to avoid coincidental high similarities of rarely visited items. This baseline is one of the most common item-to-item solutions in practical systems. This model recommends items using previous item interactions of users. The method contains the following parameters: neighborhood size (50, 100, and 200) and regularization term (16, 32, and 64) that discounts the similarity of rare items [24].

- **Item2vec** [7]: A model derived from word2vec. This model uses the interaction of items as word co-occurrences. We used the skip-gram negative sampling model for training, which was conducted in Euclidean space. The method contains the following parameters: vector dimension, learning rate, negative sampling size, and window size (3, 7, 12, and 15) [12].

- **GRU4REC** [37]: A sequence prediction model can predict the next item using recurrent neural networks when given the items a user has interacted with in the past. This model can capture long-term user preferences, unlike the other baseline models. The method contains the following parameters: the parameter vector dimension, learning rate, and batch size [37].

- **Neural collaborative filtering (NCF)** [35]: A well-known recommendation algorithm that generalizes the matrix factorization problem with a multi-layer perceptron. The method contains the following parameters: vector dimension, hidden layers for MLP, learning rate, and batch size. We set low vector dimensions (8 and 16) to avoid overfitting and employed three hidden layers for MLP ($[32, 16, 8]$ and $[64, 32, 16]$) as described in Ref.[35].

- **Light graph convolution network (LightGCN)** [34]: A simple, linear, and neat graph convolution network (GCN) model for recommendation. The method contains the following parameters: vector dimension, layers (1, 2, and 3), learning rate, and batch size.
- **Poincaré** [57]: We used the original poincaré model to get item embeddings. Because of the time and computational resource limitations of real-time stream recommender systems, we cannot embed users when there are much more users than items. Thus, we did not use the other hyperbolic embedding models [29, 79]. We ranked the items by the distance from the second latest item the user previously clicked. The method contains the following parameters: vector dimension, learning rate, negative sampling size. We set these parameters the same as common parameter settings. We used the burn-in process for five epochs at the beginning of the learning phase. This burn-in technique was introduced in Ref.[57] to get a good initial angular layout that can be helpful to find good embeddings.
- **Poincaré with regularization**: We utilize the regularization to the Poincaré method in addition to the baseline settings. Hyper-parameter of proposed graph-centralized regularization term h is $\in \{0.5, 0.75, 1\}$ in Poincaré embedding method.

Results

Table 3.3 shows the evaluation accuracy of nDCG@10 and HR@10. For each of the obtained results shown in Table 3.3, the best result is in boldface, while the best in the baselines is underlined. Overall, the hyperbolic embedding model with our regularization outperformed the other baseline methods in both nDCG@10 and HR@10. Regarding the baselines, the hyperbolic method without our regularization outperformed the other methods in nDCG@10. LightGCN also performed well in Amazon datasets. In contrast to the LightGCN results, NCF worked well for Lucra with a dense graph structure. Item-KNN performed well in nDCG@10 about the MovieLens and Online Retail dataset. As reported in Ref.[24], a deep neural network model does not always have more permanence, and a tuned item-KNN model worked well.

To break down the regularization effects, we also show the results of the standard L2 regularization and graph-centralized L2 regularization in Table 3.3. We can see that the graph-centralized regularization leveraging hierarchical graph structures had more uplift in accuracy up to 8.5%, than the standard L2 regularization and had 9% more uplift than without regularization in HR@10. In the beginning of the learning phase, we used the burn-in technique [57] to improve the angular layout without moving too far towards the boundary. This technique might have a regularization effect. Even though we used the burn-in process, the evaluation accuracy could be further improved by the regularization. This means that our proposed regularization had another effect on accuracy in addition to the burn-in technique.

To see the effectiveness of extremely low-dimensional hyperbolic embedding, we also explicitly describe the result of 10-dimensional hyperbolic embedding in Table 3.3. Despite the low embedding dimension (10 dimensions), graph-centralized regularization can maintain the hierarchical structure in the embedding space and showed that high accuracy compared with other baselines.

Figure 3.4 shows the 2D-item embedding visualization with and without regularization. We plotted the top 25% high-degree nodes as red dots and the remainder as blue dots. Using graph-centralized regularization, the high-degree nodes tended to be closer to the origin, while the low-degree nodes tended to be far from the origin. Without regularization, we see that the items tended to be embedded in the Poincaré ball's boundary. These results demonstrate that we can maintain the graph structure when we use regularization.

Regarding **RQ1**, the hyperbolic embedding model with our regularization outperformed the other baseline methods in both nDCG@10 and HR@10 and effect was more positive than standard L2 regularization.

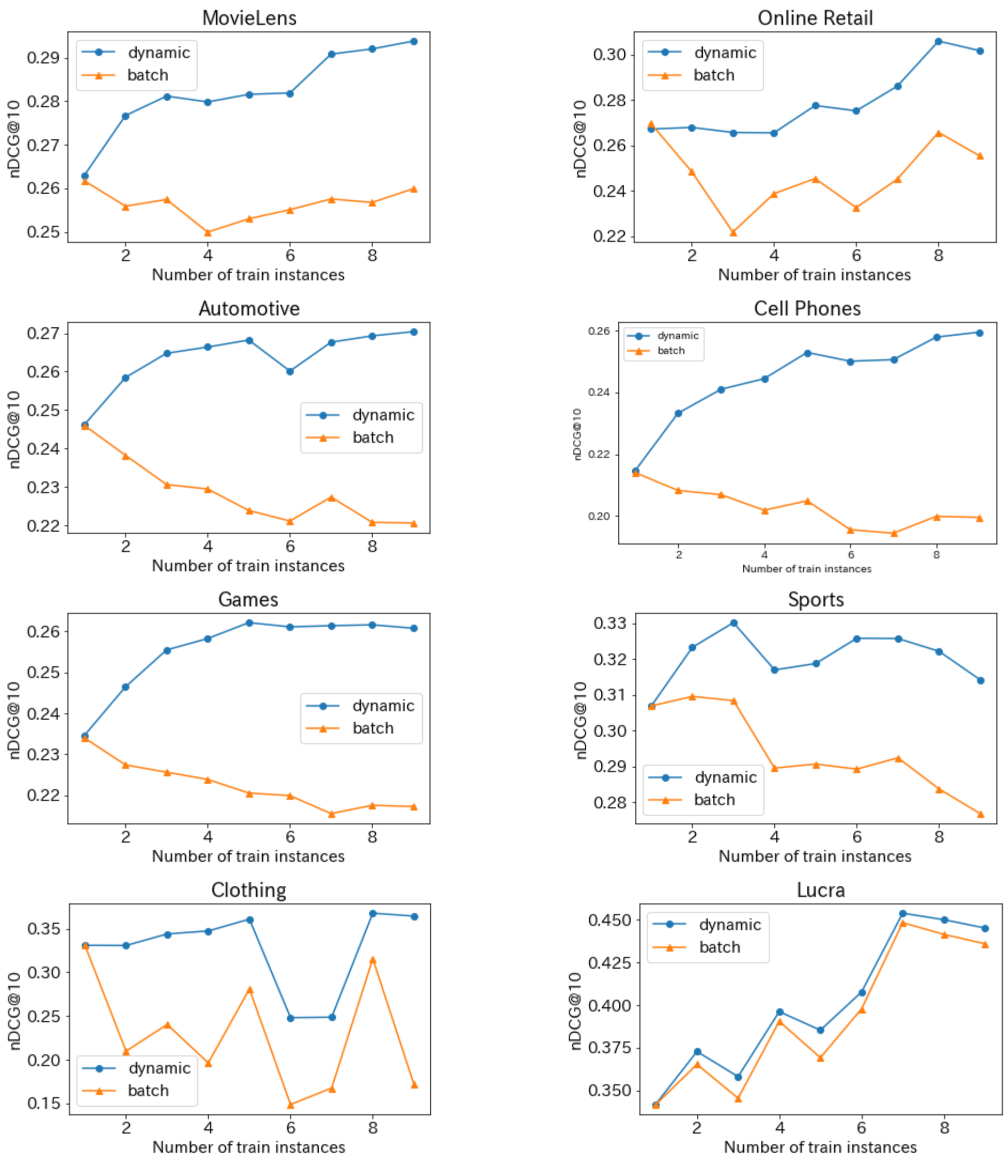

Figure 3.5: The efficiency of the dynamic and batch model updates in stream settings.

3.5.3 Efficiency Evaluation

Settings

To simulate a real-time stream recommender setting, we generated a dynamic data stream. We first sorted all of the data by timestamp and split items into 20 sets.

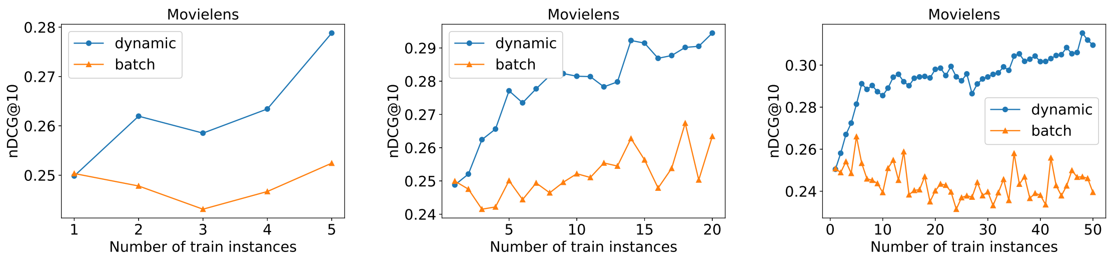

Figure 3.6: The difference between the number of train instances. Each figure left to right corresponds to the number of train instances 5, 20 and 50.

Next, we used each split set for training. Assuming many new items are added sequentially at short intervals, we trained the model by one epoch for each data stream. Thus, we did not use the burn-in process [57] in the Poincaré embedding in this efficiency evaluation setting. We first set the window size τ to 10, started building trained data from time $t = 1$, and trained from time τ to $2 * \tau$. In each testing phase for time τ to $2 * \tau$, we filtered the ground truth items to be included in the current data stream. The ground truth was re-ranked with 20 negative sampling items; the performance was judged based on the ranked list. We measured nDCG@10 for ranking metrics averaged over 20 evaluations.

Results

Figure 3.5 shows the performance of the dynamic update model strategy, where the x-axis represents the number of input stream datasets, and the y-axis represents the nDCG@10. The dynamic model outperformed batch training on all datasets. By using dynamic updates, the training proceeded more efficiently and achieved higher accuracy. The difference in improvement was small in Lucra probably because of the density of the Lucra, which tended to have more edges in the streaming data, and one epoch was enough to learn the embeddings.

Figure 3.6 shows the results of Movielens dataset for $\tau \in \{5, 20, 50\}$ and the dynamic model outperformed batch training on all parameters τ. τ controls how frequently the model is updated, and the model is updated more frequently when τ is larger. We can see the prediction performance was higher at the early stage at $\tau = 50$ while the prediction performance was improved slowly at $\tau = 5$. These results suggest that the dynamic method updatings should be as frequent as the

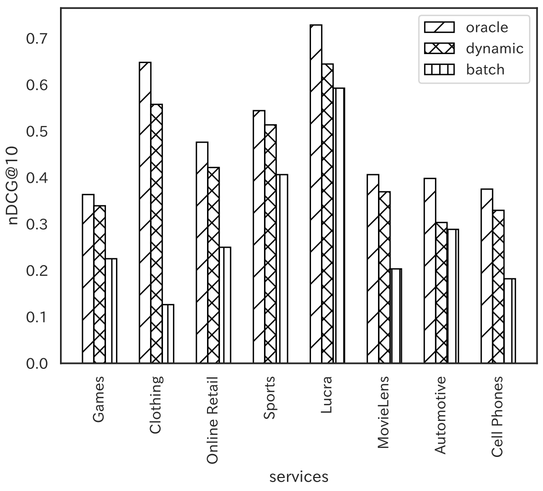

Figure 3.7: The robustness of *oracle*, batch, and dynamic model updates in setting ($T = 20$).

computational resources allow.

Figure 3.7 shows the results for robustness. The *oracle* model used all the data to train the embeddings, while the dynamic and batch models trained the embeddings in a window size of $\tau = 10$. After finishing all the training phases, we used the latest embeddings to measure the nDCG@10. The *oracle* model was the highest-scoring, followed by the dynamic model. The dynamic model had much better accuracy than the batch model due to its dynamic update embeddings and successfully maintained the embeddings' consistency even if they were not contained in the training data. We note that the *oracle* mode needed t/τ times the computational cost to get the results for each time t.

Therefore, **RQ2** can be answered as follows: The dynamic model update could achieve much better efficiency and robustness than the batch training.

3.6 Online Experiment

This section reports the results of the A/B testing. We assessed the impact on a real-world news service when the proposed method was deployed. We aimed to improve the online metrics by building compact hyperbolic embedding into the current recommender system.

3.6.1 Experimental Setup

From February 5, 2020 to February 10, 2020, we conducted A/B testing on the news service Lucra. We applied 5% traffic to the proposed method so as not to degrade the user experience.

We deployed our method to a *related article component.* This component displays related articles after a user reads an article. The existing logic for finding related articles uses title similarity and the click-through rate of the article. In this testing, we referred to the existing logic as the control and the logic using similarity of Poincaré embedding in addition to existing logic as the treatment.

We trained 15 epochs every 15 minutes while the user interactions accumulated sequentially with the 10-dimension Poincaré embedding. We set τ at three days to retain embeddings in the model. In Lucra, approximately 10,000 articles emerged in one day, and all of the articles from the three days were candidates for the recommendation.

3.6.2 Online Metrics

We used four online metrics to evaluate the impact of the proposed methods [59].

Clicks The total number of user clicks.

Click-Through Rate The total number of user clicks divided by the number of user article views. If this value is low, the user is more engaged in searching articles than browsing articles.

Clicks/Unique Users The total number of user clicks divided by the number of unique users. It is divided by the number of unique users to remove the bias of the number of users.

Dwell Time/Unique Users Total reading time of the user's related articles divided by the number of unique users. The article viewing total time refers to the time spent reading an article after clicking on a related article.

Table 3.4: The A/B testing result.

Metrics	Percent Lift
Clicks	+3.79%
Click-Through Rate	+2.58%
Clicks/Unique Users	+3.92%
Dwell Time/Unique Users	+10.2%

3.6.3 Results

Table 3.4 provides a summary of the A/B test results, showing that the treatment bucket value was significantly improved for all metrics. Not only did the number of clicks increase, but dwell time per user also improved significantly. The difference came from whether we used CF-based Poincaré embeddings. The control bucket used only the title and click-through rate, while in the treatment bucket with CF-based Poincaré embeddings, there was an increase in average dwell time. Due to the Poincaré embeddings, the articles that did not contain similar titles but would be pertinent for the user became easier to recommend. As a result, the average article dwell time increased. In the treatment bucket, 96.34% of responses could be returned with trained Poincaré embeddings. In this experiment, the Poincaré embedding was 10-dimensional. Therefore, there was no degradation of system performance when we conducted A/B testing. Our response time was kept less than 50 milliseconds. Thus, Poincaré embedding is promising as a recommender component that requires few resources.

3.7 Summary

One of the key challenges of a real-time stream recommender system is how to learn compact and accurate embedding representations efficiently. In this work, we first determined that the graphs of various recommendation datasets had a hierarchical or tree-like structure suitable for hyperbolic embedding. Then, we proposed a regularization method that uses graph-centrality. Next, we proposed a model update technique for learning hyperbolic embeddings by online learning rather than by batch learning.

In the offline evaluation, we confirmed that the regularization leveraging of the hierarchical or tree-like structures improved HR@10 up to 9% when compared with a hyperbolic embedding method without regularization. In the offline experiments in real-time stream recommender settings, we confirmed that the proposed model update technique achieved more efficient and robust updates than batch updates. Finally, we deployed our method to a million-scale news recommendation service for A/B testing. The results showed that the number of clicks improved (+3.7%) and the dwell time also improved (+10%) when *10-dimension* hyperbolic embeddings were used.

Chapter 4

An Interleaving Method for Evaluating Post-click Metrics

4.1 Introduction

Online controlled experiments are conducted daily to evaluate search and recommendation algorithms. A/B testing is a common approach that compares two different outcomes by showing them to two different user groups. A/B testing is applicable for a variety of purposes, including measurement of click-based metrics (e.g., click-through rate (CTR)), and measurement of *post-click metrics* (e.g., music listening time [30], news reading time [60, 89], and the number of reservations [32]). As post-click metrics are closely related to user satisfaction and the sales of services, the measurement of post-click metrics is particularly important for the continuous improvement of algorithms in search and recommender systems.

However, there are some challenges in measuring post-click metrics in online experiments. Suppose that a news article is ranked at the bottom of a ranking, for which users spend a significantly different length of time to read. Generally, low-ranked items are infrequently clicked. Thus, the post-click metric results in high variance of the sample mean. Moreover, some types of post-click metrics highly depending on users are of high *population* variance by nature, and also likely to result in a high variance of the sample mean. High variance in each item naturally leads to high variance for the whole ranking. This high variance can prevent online experiments from efficiently discriminating between competitive rankings.

This study proposes an efficient method for comparing the multiple rankings of post-click metrics in online experiments. The key ideas of the proposed method are (1) the decomposition of the post-click metric measurement of a ranking into a click model estimation and post-click metric measurement of each item in the ranking, and (2) interleaving multiple rankings to produce a single ranking that preferentially exposes items with high population variances. The decomposition of the post-click metric measurement permits free layout of items in a ranking and focus on the measurement of the post-click metric of each item in multiple rankings. The interleaving of multiple rankings reduces the variance of the sample mean for items with a high population variance by optimizing the ranking to be presented so that those items received more samples of the post-click metric. In this paper, the method is referred to as the *Decomposition and Interleaving for Reducing the Variance of post-click metrics* (*DIRV*). DIRV uses interleaving to evaluate post-click metrics for the variance reduction, which is a major distinction from interleaving designed to evaluate click-based metrics.

In addition to the proposal of DIRV, we have the following theoretical and technical contributions in this work. First, we proved that the ranking optimization by the DIRV leads to minimization of the evaluation error in the ranking comparison. Second, we propose two techniques to stabilize the evaluation by the DIRV. The first technique predicts the population variance of each item based on the observed samples and item features for stabilizing the ranking optimization at the beginning of the online experiment. The second technique corrects the systematic error between the estimated post-click metrics and the ground-truth post-click metrics.

We performed comprehensive experiments in simulation as well as real service settings. The experimental results revealed that: (1) the proposed method outperformed existing methods in terms of efficiency and accuracy, and the performance was especially remarkable when the input rankings shared many items, (2) the two stabilization techniques successfully improved the evaluation accuracy and efficiency.

The major contributions of this study are as follows:

- To efficiently compare post-click metrics of multiple rankings, we proposed an interleaving method (DIRV) that decomposes the post-click metric measurement and preferentially exposes items with high population variance to minimize the evaluation error.

- We provided a theoretical guarantee that the DIRV ranking optimization minimizes the evaluation error in the ranking comparison.
- We proposed two techniques to stabilize the evaluation by DIRV and demonstrated that these techniques were empirically effective.
- We extensively evaluated DIRV using both simulation and real service settings. The results demonstrated its high accuracy and efficiency.
- We published the real service data used in our experiments. This data can be used to validate our study and be used in future research on user modeling and bandit algorithms.

The remainder of this chapter is organized as follows: Section 4.2 describes the problem setting in this study. Section 4.3 explains the proposed method. Sections 4.4 and 4.5 report the experimental settings and discuss the experimental results, respectively. Finally, Section 4.6 presents the summary and future work on this topic.

4.2 Problem Setting

The primary goal of this study is to compare multiple *input rankings* $R = \{r_1, r_2, \ldots\}$ in terms of a specific post-click metric in an online experiment. A user is shown a ranking in response to her/his query, clicks the items in the ranking, and consumes the clicked items. The post-click metric of an item can only be measured after the item has been clicked by the user. To compare rankings, we define the post-click metric of a ranking as the expectation of the post-click metric of items in ranking r_i:

$$E[x|r_i] = \int_x xP(x|r_i)dx \tag{4.1}$$

where x is a random variable representing a value of the post-click metric, and $P(x|r_i)$ is the probability of observing x when a user interacts with the ranking r_i. $E[x|r_i]$ can be interpreted as how effective the ranking is to trigger users' behaviors (e.g., conversion) quantified by the post-click metric. Thus, $E[x|r_i]$ can be used to evaluate rankings based on users' post-click behaviors.

Following [61, 10, 73], we set the goal of our study to efficiently estimate the pairwise preference between rankings. The pairwise preference between rankings is defined as a matrix $\mathbf{P} \in \mathbb{R}^{|R|\times|R|}$, where $P_{i,j}$ indicates the difference between the expected post-click metric for ranking r_i and $r_j \in R$, that is,

$$P_{i,j} = E[x|r_i] - E[x|r_j]. \tag{4.2}$$

A positive value of $P_{i,j}$ indicates that the ranking r_i is superior to r_j. The binary error E_{bin} is commonly used error metric to evaluate $\mathbf{P}$ [61, 10, 73]. Letting $\overline{P}_{i,j}$ be the preference estimated by a certain method, the binary error of this estimate is defined as follows:

$$E_{\text{bin}} = \frac{1}{|R|(|R|-1)} \sum_{r_i, r_j \in R} \text{sgn}(P_{r_i,r_j}) \neq \text{sgn}(\overline{P}_{r_i,r_j}), \tag{4.3}$$

where sgn($\cdot$) returns -1 for negative values, 1 for positive values, and 0 otherwise. The operator $\neq$ returns 1 whenever the signs are unequal.

A straightforward approach for estimating $P_{i,j}$ (or obtaining $\overline{P}_{i,j}$) is to conduct A/B testing with the rankings r_i and r_j, take the mean of x for each of the rankings, and then approximate $E[x|r_i]$ and $E[x|r_j]$ using these means. We discuss possible improvements over A/B testing and introduce our proposed method in the next section.

4.3 Methodology

We propose a method named *Decomposition and Interleaving for Reducing the Variance of post-click metrics* (*DIRV*) for efficient evaluation based on post-click metrics. We first discuss possible improvements over A/B testing by deeply look at the expectation of a post-click metric and introduce DIRV as our solution. We then show that the optimization by DIRV leads to the reduction of the binary error defined in Equation (4.3). Finally, we propose two techniques to stabilize evaluations by DIRV in actual applications.

4.3.1 Decomposition

We deeply look at Equation (4.1) by decomposing it with some assumptions and discuss possible improvements to reduce the evaluation error.

In Equation (4.1), the value of a post-click metric x can be observed for each clicked item in ranking r_i. Thus, $P(x|r_i)$ can be obtained by marginalizing all of the items in r_i, with the assumption that x depends solely on d, but not on the ranking r_i:

$$P(x|r_i) = \sum_{d \in r_i} P(x|d)P(c_d = 1|r_i) \tag{4.4}$$

where c_d is a random binary variable indicating an event of click on d, $P(c_d = 1|r_i)$ is the click probability of d in the ranking r_i, and $P(x|d)$ is the probability of observing x at d.

The independence assumption between x and r_i allows the following decomposition of the expectation of the post-click metric:

$$\begin{aligned} E[x|r_i] &= \int_x x \left\{ \sum_{d \in r_i} P(x|d)P(c_d = 1|r_i) \right\} dx \\ &= \sum_{d \in r_i} \int_x xP(x|d)P(c_d = 1|r_i)dx \\ &= \sum_{d \in r_i} P(c_d = 1|r_i) \int_x xP(x|d)dx \\ &= \sum_{d \in r_i} P(c_d = 1|r_i)E[x|d] \end{aligned} \tag{4.5}$$

where $E[x|d]$ is the expectation of the post-click metric for item d.

This equation immediately suggests that an unbiased estimate of $E[x|r_i]$ can be obtained by:

$$\overline{E}[x|r_i] = \sum_{d \in r_i} \overline{P}(c_d = 1|r_i)\overline{E}[x|d] \tag{4.6}$$

where $\overline{P}(c_d = 1|r_i)$ and $\overline{E}[x|d]$ are unbiased estimates for $P(c_d = 1|r_i)$ and $E[x|d]$, respectively. A standard approach to obtain these two estimates is to use the sample means of the random variables c_d and x for $\overline{P}(c_d = 1|r_i)$ and $\overline{E}[x|d]$, respectively. This decomposition can potentially achieve higher efficiency than A/B testing, because the samples for item d in *any* rankings in R can be used to estimate $E[x|d]$. Thus, we can increase the sample size for items that are shared by multiple rankings. Rankings are likely to include identical items, especially when differences between similar rankings (e.g., the same algorithm with different parameters) are evaluated.

While only Equation (4.6) enables better estimation of $E[x|d]$ by increasing the sample size, the use of the sample mean is not efficient enough especially when (1) the population variance of x is high and (2) the sample size of x is small due to small $P(c_d = 1|r_i)$. Both cases are likely to lead to high variance of $\overline{E}[x|d]$, resulting in high variance of $\overline{E}[x|r_i]$. Consequently, a large binary error in the ranking comparison is obtained. Both cases can be alleviated by prioritizing the exposure of items that satisfy these conditions for increasing the sample size. However, simple manipulation of a ranking prevents us from correctly estimating $P(c_d = 1|r_i)$ since this probability depends on the position of item d in ranking r_i.

Hence, we introduce a click model for further decomposing $\overline{P}(c_d = 1|r_i)\overline{E}[x|d]$ in the summation. We assume the examination hypothesis [20] that the item click can be decomposed into two variables: examination e_d and attraction a_d. This enables us to decompose the unbiased estimate $\overline{P}(c_d = 1|r_i)$ as follows:

$$\overline{P}(c_d = 1|r_i) = \overline{P}(e_d = 1|r_i)\overline{P}(a_d = 1|d), \tag{4.7}$$

where $\overline{P}(e_d = 1|r_i)$ can be estimated by a position-based click model or a cascade click model [20, 23]. The position-based click model assumes that the examination probability depends solely on the rank of the item in ranking r: i.e., $\overline{P}(e_d = 1|r_i) = g(\text{rank}(d, r))$, where g is a function taking a rank to return a probability, and rank(d, r) is the rank of item d in ranking r. This assumption allows us to avoid estimating the probabilities specific to a particular ranking: i.e., $\overline{P}(e_d = 1|r_i)$. The estimations of $E[x|d]$, $P(e_d = 1|r_i)$ and $P(a_d = 1|r_i)$ are detailed in Section 4.4.3. Note that the simple assumption for the click model might sacrifice the evaluation error for efficiency. Thus, in Section 4.3.4, we introduce a technique correcting the systematic error between the assumed and actual click models.

In summary, the decomposition of Equation (4.1) increases the sample size of x for estimating $E[x|d]$ and provides greater flexibility about the ranking presented to the users. Now one can freely produce and present a ranking containing items from input rankings R, and estimate each of $P(a_d = 1|d)$ and $E[x|d]$ based on the users' interactions with the presented ranking, in order to obtain $\overline{E}[x|r_i]$. In the next subsection, we explain what ranking should be presented to the users for minimizing the evaluation error.

4.3.2 Interleaving

Our proposed DIRV method is designed to minimize the variance of a post-click metric by interleaving input rankings to produce a single ranking to be presented to the users. The interleaved ranking preferentially exposes items with high population variance. The method attempts to receive more samples (or clicks) for these items to reduce the variance of the input rankings.

As shown in Appendix 6.2, the variance $V[\overline{E}[x|r_i]]$ can be defined as a monotonically decreasing function ϕ on the number of impressions of the item d (denoted by n^i_d) and the number of clicks on the item d (denoted by n^c_d):

$$V[\overline{E}[x|r_i]] = \sum_{d \in r_i} \phi_{d,r_i}(n^i_d, n^c_d). \tag{4.8}$$

Our proposed DIRV method determines a ranking o produced by interleaving input rankings. The ranking o minimizes the summation of $V[\overline{E}[x|r_i]]$ over all the rankings in R when o is exposed to a user:

$$\min_o f(o) \tag{4.9}$$

where

$$f(o) = \sum_{r_i \in R} V[\overline{E}[x|r_i]] = \sum_{r_i \in R} \sum_{d \in r_i} \phi_{d,r_i}(\dot{n}^i_d, \dot{n}^c_d), \tag{4.10}$$

$$\dot{n}^i_d = n^i_d + 1, \qquad \dot{n}^c_d = n^c_d + E[c_d|o]. \tag{4.11}$$

$\dot{n}_d$ indicates the expected sample size after the interleaved ranking o is presented to a user. The number of impressions, n^i_d, is incremented by one each time the ranking o is presented. The number of clicks for item d, n^c_d, is incremented by $E[c_d|o] = P(c_d = 1|o)$, which indicates the expected number of clicks on item d in ranking o for a single impression.

As a brute-force search is infeasible to determine the optimal ranking o, we employ a greedy algorithm based on Equation (4.9). The greedy algorithm starts with an empty ranking r, and repeats appending an item to r that maximizes the difference between the current ranking and the ranking with the new item in terms of the variance of the sample mean:

$$\max_{d \in D \setminus r} \sum_{r_i \in R} \left(\phi_{d,r_i}(n^i_d, n^c_d) - \phi_{d,r_i}(\ddot{n}^i_d, \ddot{n}^c_d) \right), \tag{4.12}$$

where

$$\ddot{n}^i_d = n^i_d + 1, \quad \ddot{n}^c_d = n^c_d + E[c_d | r \oplus d]. \tag{4.13}$$

and D is a set of all items in R and $r \oplus d$ is the ranking r with d appended to the bottom. This greedy algorithm repeatedly finds the item to minimize the variance when it is appended. The algorithm stops when the ranking reaches a predefined depth.

4.3.3 Theoretical Justification

We provide theoretical justification that the minimization of the summation of $V[\overline{E}[x|r_i]]$ over R leads to the minimization of the expected binary error $E[E_{\text{bin}}]$ defined in Equation (4.3).

Theorem 4.3.1.

$$\sum_{r_i \in R} V[\overline{E}[x|r_i]] \tag{4.14}$$

constitutes the upper bound of $E[E_{\text{bin}}]$.

Proof. Let $\mu_i = E[x|r_i]$ and $\overline{\mu}_i = \overline{E}[x|r_i]$. Without a loss of generality, we assume that $\mu_i > \mu_j$. The binary error probability $P(\text{sgn}(P_{i,j}) \neq \text{sgn}(\overline{P}_{i,j}))$ can be interpreted as the probability of estimating a higher value for μ_j than that for μ_i, that is, $P(\text{sgn}(P_{i,j}) \neq \text{sgn}(\overline{P}_{i,j})) = P(\overline{\mu}_j > \overline{\mu}_i) = P(\overline{\mu}_j - \overline{\mu}_i > 0)$. Letting $\Delta_{ij} = \mu_j - \mu_i$ and $\overline{\Delta}_{ij} = \overline{\mu}_j - \overline{\mu}_i$, and using Chebyshev's inequality $(P(|x - \mu| > k) \leq \sigma^2/k^2)$, this probability can be bounded as follows:

$$\begin{aligned} P(\text{sgn}(P_{i,j}) \neq \text{sgn}(\overline{P}_{i,j})) &= P(\overline{\Delta}_{ij} - \Delta_{ij} > -\Delta_{ij}) \\ &\leq P(|\overline{\Delta}_{ij} - \Delta_{ij}| > -\Delta_{ij}) \\ &\leq \frac{V[\overline{\Delta}_{ij}]}{\Delta_{ij}^2} = \frac{V[\overline{\mu}_i] + V[\overline{\mu}_j]}{\Delta_{ij}^2} \end{aligned} \tag{4.15}$$

Note that $P(|x| > y) = P(x > y) + P(x < -y) \geq P(x > y)$, and $V[x - y] = V[x] + V[y]$ if x and y are assumed independent.

Therefore, the expected binary error is bounded as follows:

$$\begin{aligned} E[E_{\text{bin}}] &= \frac{1}{|R|(|R|-1)} \sum_{r_i, r_j \in R} P(\text{sgn}(P_{i,j}) \neq \text{sgn}(\overline{P}_{i,j})) \\ &\leq \frac{1}{|R|(|R|-1)} \sum_{r_i, r_j \in R} \frac{V[\overline{\mu}_i] + V[\overline{\mu}_j]}{\Delta_{ij}^2} \\ &\leq \frac{1}{C|R|} \sum_{r_i \in R} V[\overline{\mu}_i] = \frac{1}{C|R|} \sum_{r_i \in R} V[\overline{E}[x|r_i]] \end{aligned} \tag{4.16}$$

where C is the smallest value for Δ_{ij}^2 $(r_i, r_j \in R)$. Note that $V[\overline{\mu}_i]$ appears $|R| - 1$ times in the summation, which cancels out $|R| - 1$ in the denominator. □

This theorem suggests that the upper bound of $E[E_{\text{bin}}]$ becomes lower if the variance of $\overline{E}[x|r_i]$ is reduced. This is the theoretical basis of our proposal and is formally expressed as follows:

$$E[E_{\text{bin}}] \to 0 \quad \left(\sum_{r_i \in R} V[\overline{E}[x|r_i]] \to 0 \right). \tag{4.17}$$

4.3.4 Stabilization Techniques

There still remain some practical problems regarding the variance reduction and systematic error associated with applying DIRV to an online experiment. These problems are discussed in this section.

Feature-based Variance Prediction

The first problem is the variance estimation of the post-click metric $V[x|d]$. This variance is included in the variance of $\overline{E}[x|d]$ (see Appendix 6.2) and is required to be estimated to produce interleaved rankings. However, the estimation can be inaccurate, especially when the sample size is small. As demonstrated in our experiments, this problem can cause degradation of the estimation efficiency.

To overcome this problem, we propose using a regression model that predicts the variance of the post-click metric based on item features (e.g., titles and categories). As a result, we can use two estimates for the variance of the post-click metric $V[x|d]$, namely, the unbiased variance obtained from the observed values (denoted by $\overline{V}[x|d]$), and the predicted variance estimated by a regression model (denoted by $\hat{V}[x|d]$).

The larger value between $\overline{V}[x|d]$ and $\hat{V}[x|d]$ (i.e., $\max(\overline{V}[x|d], \hat{V}[x|d])$), is used as the estimate of $V[x|d]$. This idea is similar to the *clipping technique* used in [17, 31, 77]. As DIRV prioritizes items with a high variance in the interleaved ranking, there are opportunities to correct the estimation errors for items whose variance of the post-click are overestimated. In contrast, underestimation of the variance is more problematic because there are only limited opportunities to correct the error. Therefore, we use a higher variance to avoid underestimating the variance. Our experiments demonstrated that this technique is effective for efficient evaluation.

Systematic Error Correction

The second technique reduces the systematic error between the actual and assumed click models. Although the click probability $P(c_d = 1|r_i)$ is efficiently estimated with the click model, its estimation accuracy may not be sufficient as the click model does not precisely reflect real user behaviors.

To alleviate this problem, we estimate the click probability $P(c_d = 1|r_i)$ by the click model as well as a model-agnostic estimate (i.e., the sample mean of clicks). We present each ranking r_i with a small probability and observe the clicks on d in r_i. The clickthrough rate (i.e., $\hat{P}(c_d = 1|r_i) = n^c_{d,r_i}/n^i_{r_i}$) is used together with $\overline{P}(c_d = 1|r_i)$ in Equation (4.7), where n^c_{d,r_i} is the number of clicks on item d in the ranking r_i and $n^i_{r_i}$ is the number of impressions of the ranking r_i. The two estimates are linearly combined for approximating $P(c_d = 1|r_i)$:

$$P(c_d = 1|r_i) \approx \theta_i \overline{P}(c_d = 1|r_i) + (1 - \theta_i)\hat{P}(c_d = 1|r_i) \tag{4.18}$$

where we set the parameter θ_i to a value decreasing as the sample size increases (i.e., $1/(n^i_{r_i} + 1)^{0.5}$) to weight the click model when $n^i_{r_i}$ is small.

To present each ranking r_i with a small probability, we incorporate another objective function $g(o)$ into the policy for generating rankings. $g(o)$ is defined as follows:

$$g(o) = \sum_{r_i \in R} \theta_i \sum_{d \in r_i} \phi_{d,r_i}(\dot{n}^i_{r_i}, \dot{n}^c_{d,r_i}) \tag{4.19}$$

where

$$\dot{n}^{i}_{r_i} = n^{i}_{r_i} + \mathrm{sgn}(r_i = o), \quad \dot{n}^{c}_{d,r_i} = n^{c}_{d,r_i} + \mathrm{sgn}(r_i = o)P(c_d = 1|o) \tag{4.20}$$

and $\mathrm{sgn}(r_i = o)$ returns 1 if all of the items are identical in r_i and o, otherwise 0. In the last procedure for generating rankings, we select the ranking o that minimizes $f(o) + \gamma g(o)$ from the set $\{o^*\} \cup R$, where o^* is a greedy solution in Equation (4.9), and γ is a hyper-parameter. Although this technique is not justified theoretically, it was shown to be effective by our experiments.

4.4 Experiment Setup

We describe the experimental settings to answer the following research questions (RQs):

- **RQ1**: Can DIRV identify preferences between rankings more accurately and efficiently than other methods?
- **RQ2**: How does the variance prediction technique affect the evaluation efficiency?
- **RQ3**: How does the error correction technique affect the evaluation accuracy?

We have two experimental settings for comprehensive validation, namely, the simulation-based setting and the real service setting. A summary of the experimental settings is included in Table 4.1. The simulation-based setting is based on the traditional setting but is extended for the post-click evaluation. The real service setting is designed to validate methods close to the actual online evaluation by use of raw user feedback about the clicks and post-click values obtained from the raw ranking impressions. The real service dataset[1] are described in Section 4.4.2.

[1]It is public available at https://github.com/koiizukag/DIRV.

Table 4.1: Experimental Setting Summary

	Simulation-based	Real Service
Input ranking	generated	raw data
Systematic error	-	exists
Click	partially generated	raw data
Post-click	partially generated	raw data

4.4.1 Simulation-based Settings

Evaluation Procedure

With the exception of post-click behavior, we simulated user behavior using the same four steps as in the previous interleaving evaluations [66, 75]:

1. **Impression**: A list of ranked items is displayed for a user.
2. **Click**: The user decides whether to click an item.
3. **Post-click Behavior**: If the user clicks the item, then the user consumes it, and post-click values are produced.
4. **Session End**: The user ends the session.

The details of the user's behaviors are as follows:

First, we fix a query by uniformly sampling from a static dataset. Then, each method generates rankings from the query and displays them to the users. Whether a user clicks on an item depends on the item's click probability.

After clicking on the item, the user is assumed to consume the item. Then, we can measure the value of a post-click metric (e.g., dwell time). The post-click value is assumed to follow a particular distribution. For example, the occurrence of a conversion in EC dataset follows a Bernoulli distribution. Throughout the experiment, we assumed a cascade click model [20], in which users view results from top to bottom and leave as soon as they see a worthwhile item. Each experiment involved a sequence of 10,000 simulated user impressions. All runs were repeated 30 times for each dataset and parameter setting.

Datasets

Three datasets were used in the simulation-based settings. The first dataset (called News) was generated from our news media service named Gunosy. The second dataset (called LETOR) is the learning to rank dataset. These two datasets were used to measure the dwell time that varies for each user. The third dataset (called EC) is an artificially generated dataset used to simulate e-commerce product purchases that has a predefined post-click value for the purchase price of the product.

News Dataset This dataset comes from a real-world, mobile news application[2]. The News dataset contains the dwell time and the click probability. We generated this dataset using the implicit feedback from a single day. Each selected article had more than 20,000 raw post-click values (i.e., the dwell time for each user). This dataset contains 50 articles, which is sufficient to generate rankings. The ground truth for click probability and dwell time for each article was calculated from all of the user logs.

LETOR Dataset We used eight publicly available LETOR datasets with varying sizes and representing different search tasks [75, 61]. Each dataset consisted of a set of queries and a set of items corresponding to each query. Feature representations and relevance labels were provided for each item and query pair. The datasets had three labels: unrelated (0), relevant (1), and highly relevant (2). The LETOR dataset was broken down by task. Most of the tasks were from the TREC Web Tracks from 2003 to 2008 [65, 21, 80]. These TREC Web Tracks were HP2003, HP2004, NP2003, NP2004, TD2003, and TD2004. Each TREC Web Track contained 50–150 queries. The OHSUMED dataset was based on a search engine's query log of the MEDLINE abstract database and contained 106 queries. MQ2007 was based on the million query track [4] and consisted of 1,700 queries.

We generated click probability and post-click values because these datasets do not contain these values. We assumed that a user would read an article for a long time if the article was relevant to the user. To reflect this, we generated dwell time values using an exponential distribution with the rate parameters set

[2]https://drive.google.com/drive/folders/1cE2VuVDP1bLg8moDrBaX0C21L-AnWKm8?usp=sharing

to $\frac{1}{\lambda_d} \sim (\text{relevance} + 1) \cdot \text{uniform}(1, 20)$ for each item. We set the click probability to $P(a_d = 1|d) \sim \min((\text{relevance} + 1) \cdot \text{uniform } (0.0, 0.5), 1)$.

EC Dataset We generated a dataset to simulate a product purchase in an e-commerce setting. This dataset can be easily reproduced based on the parameters described below. In this dataset, we randomly set the click probability to $P(a_d = 1|d) \sim$ uniform $(0.0, 0.5)$, price$(d) \sim$ uniform $(1, 1000)$, and conversion_rate$(d) \sim$ uniform $(0, 0.5)$. The relationship between the conversion rate and the price is $E[x|d] = \text{conversion\_rate}(d) \cdot \text{price}(d)$. We generated 50 items using these parameters.

Input Ranking

In the LETOR datasets, we generated *input rankings* by sorting items with features used in the existing interleaving experiments [75, 61]. We used the BM25, TF.IDF, TF, IDF, and LMIR.JM features for MQ2007. For the other datasets, we used the BM25, TF.IDF, LMIR.JM, and Hyperlink features. First, we randomly selected 20 items to obtain the same number of item candidates for each dataset. We then sorted ten items by each feature. As a result, we generated $|R| = 5$ rankings of length $|r_i| = 10$, which were similar to the settings reported in [75].

For the News and EC datasets, we generated input rankings by prioritizing items with positive feedback. We assumed that News and EC companies are likely to pay more attention to the user's feedback in their ranking algorithm. The algorithm for generating input rankings first retrieved the top-k items ranked in descending order of $P(a_d = 1|d)E[x|d]$. This algorithm was used to approximate the level of user satisfaction. Next, we randomly retrieved items that had not been selected and appended them to the input ranking. We continued this process until the ranking reached a certain length. Finally, we randomly shuffled the order of each input ranking randomly. We used different values for k in our experiments, which we called the *item duplication rate*. As a result, we generated $|R| = 5$ input rankings of length $|r_i| = 10$, which are similar values to the setting reported by [75].

Table 4.2: **The evaluation accuracy for the News, EC, and LETOR datasets in the simulation-based setting. The binary error E_{bin} of all the comparison methods after 10,000 impressions on comparisons of $|R| = 5$ rankings was averaged over 30 times per dataset and parameter. The best performances are noted in bold.**

	News					EC				
Duplication ratio (%)	0	20	40	60	80	0	20	40	60	80
A/B Testing	0.100	0.190	0.095	0.140	0.160	**0.015**	0.040	0.070	0.075	0.185
TDM	0.425	0.465	0.470	0.500	0.615	0.055	0.170	0.200	0.215	0.300
DIRV w/o Var Pred	0.150	0.280	0.115	0.100	0.110	0.350	0.280	0.275	0.320	0.335
DIRV w/o Err Corr	**0.075**	**0.095**	**0.070**	**0.035**	**0.070**	0.035	0.045	0.055	**0.030**	**0.050**
DIRV	0.095	0.165	0.080	0.045	0.075	**0.015**	**0.035**	**0.045**	0.055	**0.050**

	LETOR							
	HP2003	HP2004	TD2003	TD2004	NP2003	NP2004	MQ2007	OHSUMED
A/B Testing	0.085	0.105	0.070	0.110	0.095	0.080	0.100	0.085
TDM	0.360	0.255	0.295	0.345	0.285	0.330	0.355	0.295
DIRV w/o Var Pred	0.070	0.055	0.060	0.060	0.075	0.055	0.065	0.035
DIRV w/o Err Corr	**0.020**	0.045	**0.035**	**0.025**	**0.030**	**0.030**	**0.015**	0.040
DIRV	0.030	**0.015**	**0.035**	0.040	0.045	0.035	0.055	**0.020**

4.4.2 Real Service Settings

Simulation-based experiments are easily conducted. However, their settings are limited in the validation of their assumptions. In the proposed method, estimating the expected post-click metric relies on the assumption that post-click behavior is independent of other behaviors by the user. However, this assumption may not always be true. For example, in the news service, the dwell time on a clicked article may depend on the dwell time of another clicked article when the two articles contain similar or redundant content. The simulation-based experiments were based on user simulation, which assumes the post-click behavior is independent. Moreover, the assumed click model may have been wrong or contained estimation errors. Therefore, the validity of these assumptions was not tested by the simulation-based experiments. We generated a real service dataset using the rankings presented to real users to validate the proposed method under conditions closer to actual user behaviors in an online evaluation.

Real Service Dataset

The evaluation of interleaving requires 1) *input rankings* to be used as the input, 2) generation of *interleaved rankings* from the *input rankings* for presentation to the users, and 3) *user behaviors* associated with the interleaved rankings. The interleaving method generates interleaved rankings from the input rankings using various policies for generating the rankings. To evaluate methods in this setting, we generated interleaved rankings by identifying *all possible combinations* of the items that the input rankings could generate and collected user behavior associated with the interleaved rankings from the service log. As they cover all possible interleaved rankings, one can obtain real statistics observed for any interleaved rankings without actually presenting them.

The data were collected in a news application. The news application was the same application used in the simulation-based News dataset. This dataset consisted of a set of queries, a set of input rankings and the interleaved rankings with user behaviors corresponding to each query. User behaviors data contained click or not for each article in the ranking and dwell-time for each article that the user clicked. The query was composed of the topic and day requested by the user. We used a social topic that was one of the most popular topics in this service. The ranking was personalized and differed for each user and query pair. The top three rankings with the highest number of impressions for each day were selected as the input ranking. The rankings were fixed at three in length to reduce the possible number of item combinations, since the number of interleaved rankings could be explosively large depending on the number of items in the input rankings. We generated 30 pairs of queries and their respective responses for a million-scale ranking impression in a one-month period.

The real service setting has advantages and disadvantages over simulation-based settings. The real service dataset was limited by the number of items in the input rankings. In addition, interleaved rankings did not cover all combinations of the items in the input rankings. On the other hand, this real service setting did allow us to test the assumptions of the model close to the actual online evaluation.

Experimental Runs

The ground truth of the preference in E_{bin} was calculated from half of the total data included for each query using A/B testing logic. The method was validated using the other half of the data. The experimental runs consisted of 5,000 ranking impressions for each of the 30 queries. These procedures were iterated 30 times.

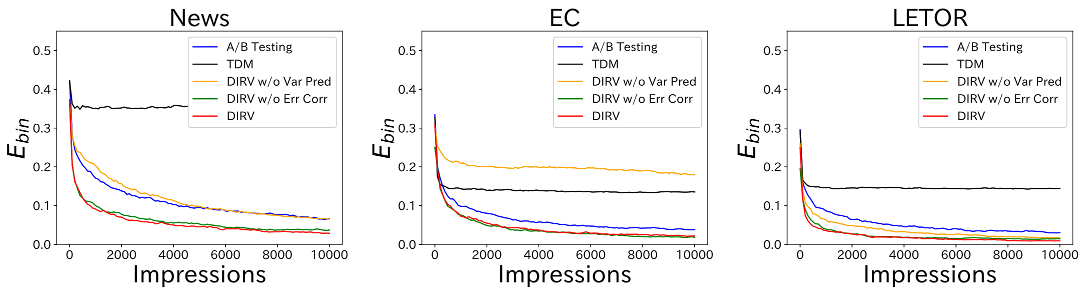

Figure 4.1: The E_{bin} averaged over the number of impressions of the News, EC, and LETOR datasets. For all the datasets, DIRV or DIRV w/o Err Corr had the lowest E_{bin} for each impression.

4.4.3 Parameter Estimation

Each parameter was estimated as follows: $E[x|d]$ was estimated by the sample mean of the observed post-click value x for item d over all rankings. $P(e_{d_j} = 1|r_i)$ that was the examination probability of j-th item in the ranking r_i was estimated assuming the cascade click model as $\overline{P}(e_{d_j} = 1|r_i) = \prod_{k=1}^{j-1}(1 - \overline{P}(c_{d_k} = 1|d_k))$ where $\overline{P}(c_{d_k} = 1|d_k) = \overline{P}(e_{d_k} = 1|d_k)\overline{P}(a_{d_k} = 1|d_k)$. $P(a_d = 1|d)$ was estimated by $\overline{P}(a_d = 1|d) = n_d^c/n_d^e$ where n_d^c is the number of clicks and n_d^e is the number of presentations of item d over all of the rankings. We note that n_d^e is incremented for the items between the 1st position and the last position of the click in the ranking for each ranking impression. If there were no clicks in the ranking, n_d^e for each item was increased.

4.4.4 Variance Prediction

We used the News and real service datasets for studying the variance prediction because the EC and LETOR datasets had no ground-truth for the variance. We generated a dataset that included each article's title length and category, as well as the media source and the sample variance for ground-truth that was calculated from the dwell time of all users. A tree-based training model with a gradient-boosting framework [3] was used for prediction. We used the features detailed in Table 4.3 for prediction. The training epoch was set to 1,000. The root mean

[3]https://github.com/microsoft/LightGBM

Table 4.3: Features and importance

Feature	Importance
Category ID to which the article belongs	879
Supplier ID of the article	2,342
Content length of the article	1,854
Title length of the article	1,045

square error was used for the loss function. The early stopping parameter was 10. The other parameters were set to default values.

Figure 4.4 shows the plots of the predicted variance and the actual variance. A Pearson correlation value of 0.76 was achieved by only using meta-features. Feature importance is shown in Table 4.3.

In the EC and LETOR datasets, we artificially generated the predicted variance. Specifically, uniformly randomized noise was added to the population variance $V[x|d]$ to generate a predicted $\hat{V}[x|d]$ that satisfies $\hat{V}[x|d] \leq 2V[x|d]$.

4.4.5 Comparison Methods

We used five methods for the evaluation: A/B Testing, modified TDM, DIRV w/o Var Pred, DIRV w/o Err Corr, and DIRV. PPM is designed to evaluate click-based metrics and considered as the state-of-the-art method for interleaving [61]. However, it is not trivial to modify PPM to evaluate post-click metrics. As previously discussed, the modified TDM [74] is the only method designed to evaluate non-click-based metrics using interleaving. Based on [74], we modified the credit function to be the product of the post-click value and the original credit function of the TDM. DIRV w/o Var Pred is a DIRV method without the variance prediction, while DIRV w/o Err Corr is a DIRV method without the error correction. DIRV is a method that has both stabilization techniques. The hyper-parameter γ was set to 1.0.

TDM could not be used in the real service setting because there are cases where the rankings generated by the TDM might not exist in the real service data, as discussed in 4.4.2. Therefore, TDM was only used for the simulation-based experiments. In the DIRV methods for the real service setting, we generated rankings that minimize variance using $f(o)$ and $g(o)$ from the candidates of the interleaved rankings.

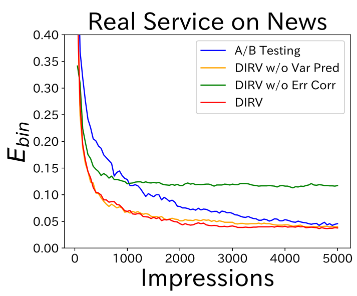

Figure 4.2: The E_{bin} averaged over the number of impressions of the real service News dataset. DIRV or DIRV w/o Var Pred had the lowest E_{bin} for each impression.

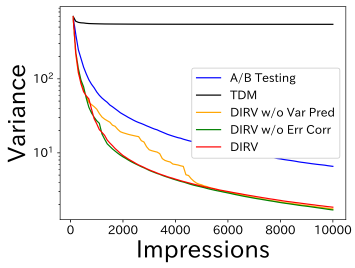

Figure 4.3: The variance versus the number of impressions in the simulation-based News dataset. The result shows that DIRV had lower variance for each impression, especially for a small number of impressions.

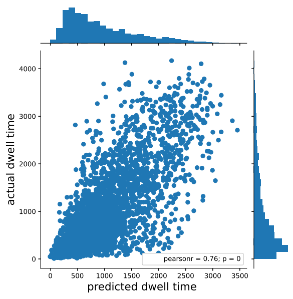

Figure 4.4: The visualization of the predicted variance. Each dot corresponds to the predicted and actual variance values for each News article. The Pearson's correlation value was 0.76.

4.5 Results and Discussion

In this section, we discuss the experimental results for answering the research questions posed in section 4.4.

4.5.1 Efficiency and Accuracy

First, we answer the RQ1: Can DIRV identify preferences between rankings more efficiently and accurately than comparison methods?

Efficiency

Figure 4.1 shows the efficiency results for the simulation-based setting for the News, LETOR, and EC datasets. Figure 4.2 shows the efficiency for the real service setting. In Figures 4.1 and 4.2, the x-axis represents the number of impressions, and the y-axis represents E_{bin}. For all of the datasets, DIRV had the lowest E_{bin} for each impression. In contrast to the simulation-based setting, DIRV w/o Err Corr had higher E_{bin} after $1,000$ impressions compared to the results from the

A/B testing in the real service setting. TDM stopped decreasing E_{bin} during the early stage in all datasets.

Among the evaluated methods, DIRV achieved the highest efficiency by reducing the variances. DIRV was designed to reduce variance by (1) aggregating post-clicks from different rankings that leads to an increased sample size for each item and (2) exposing the item that has high variance. DIRV successfully reduced the variance, which led to high efficiency (as shown in Figure 4.3).

Accuracy

Table 4.2 details the evaluation accuracy results. The number closest to 0.0 for each parameter and dataset are highlighted in bold. In the LETOR dataset, there were eight E_{bin} results for each dataset. In the News and EC datasets, there were five E_{bin} results based on the item duplication ratios (which ranged from 0% to 80%, in 20% increments). Figure 4.2 shows the accuracy of the real service setting in the last impression (i.e., at 5,000 impressions).

Overall, DIRV outperformed the existing methods for all of the datasets. A/B testing resulted in the lowest E_{bin} in the EC dataset with duplication ratios of 0%. We designed the estimator using the expectation for post-click metrics. DIRV achieved high accuracy compared with TDM due to the use of our estimator. It is remarkable that our method performed well when many items were shared among the input rankings (i.e., high duplication ratio). Similar to the results from [74], our results demonstrated that TDM is difficult to extend for aggregating continuous values like post-click values.

Regarding **RQ1**, DIRV outperformed the existing methods in efficiency and accuracy. The performance was especially remarkable when many items were shared among the input rankings.

4.5.2 Variance Prediction Technique

Next, we answer the RQ2: How does the variance prediction technique affect the evaluation efficiency? Figure 4.3 details the variance reduction on the News dataset in the simulation setting. The results show that DIRV achieved the lowest variance among the evaluated methods. DIRV w/o Var Pred decreased the variance slowly compared to DIRV, especially for a small number of impressions. TDM had the highest variance for each impression. These trends were also observed in the other datasets.

Figure 4.3 shows that the predicted variance contributed to reducing the variance in small impressions. In contrast, the variance of some items was underestimated in DIRV w/o Var Pred and could not obtain a sufficient number of exposures during the early stage. This resulted in reducing the variance in the post-click metrics slowly. The variance prediction technique reduced the variance and improved efficiency. In TDM, some items at the bottom of the input ranking was not selected for the interleaved ranking, which led to high variance.

Figure 4.2 shows that the variance prediction did not contribute to the efficiency in the real service setting. This is because the total number of items was small in this setting. Thus, there were fewer benefits from manipulating the order of the items using the predicted variance.

Regarding **RQ2**, the variance prediction techniques contributed to reducing the variance, which led to improved efficiency.

4.5.3 Error Correction Technique

Finally, we answer the RQ3: How does the error correction technique affect the evaluation accuracy? Figure 4.2 shows the accuracy of the real service setting at the end of the evaluation (i.e., at 5,000 impressions). The results show that the accuracy was close to 0.0 for DIRV and A/B testing methods. In contrast, the DIRV w/o Err Corr method stopped decreasing the E_{bin} after 1,000 impressions.

The results show that the error was reduced as the number of impressions increased in the real service setting by introducing an error correction technique. In particular, we successfully confirmed that the evaluation error converged around at 0.0 using the error correction technique, which reduced the systematic error as the experiment progressed. This result supported the assumption that the post-click behavior was independent of other user behavior in this real service setting.

The difference between the assumed cascade click model and the actual user behavior in the real service setting resulted in a systematic error. Figure 4.2 illustrated the error in the results of the DIRV w/o Err Corr method. In the simulation-based setting, we assumed that the actual user behavior also obeyed the cascade click model. This assumption led to the same accuracy between the

DIRV and DIRV w/o Err Corr methods in Table 4.2.

Regarding **RQ3**, we reduced the systematic error using the error correction technique. The error correction technique led to improved accuracy when an estimation error existed in the click model.

4.6 Summary

In this study, we proposed a method for accurate and efficient post-click evaluation using interleaving. First, we introduced the click model to aggregate the post-click values. Then, we showed that minimizing the variance of post-click metrics leads to a reduction in the evaluation error. Next, we provided a policy to generate rankings to reduce the variance. Finally, we proposed two stabilization techniques to support the proposed method.

To evaluate this method, we conducted comprehensive experiments with both simulation-based and real service settings. The experimental results from this study indicated that 1) the proposed method outperformed existing methods in efficiency and accuracy and the performance was especially remarkable when many items were shared among the input rankings, 2) the variance prediction techniques contributed to reducing the variance, which led to improved efficiency, and 3) we could successfully reduce the systematic error using the error correction technique, which led to improved accuracy when there existed an estimation error in the click model.

Chapter 5

Evaluating the Effects of News Articles on Ad Consumption

5.1 Introduction

News recommendation systems have played an important role in satisfying users' information needs, as a large amount of news items are produced daily on the Web, which prevents users from finding information of their interests. It is necessary for news providers to present news articles that satisfy users and, at the same time, generate revenue to deliver service value to users sustainably. Thus, practical news services should provide users with rankings consisting of both news articles and advertisements (e.g., products, services, and information) to satisfy users and generate revenue.

The position of the news articles and ads has been optimized based on user behaviors such as click behaviors, with a strong assumption that articles and ads are independent from each other when applied to real world services [91, 88]. However, as we show in this work, user behaviors toward ads are affected by news articles, and, accordingly, they need to be arranged with careful consideration of their mutual influence.

In this paper, we investigate the effect of news articles on ads and show that the ad effectiveness depends significantly on the news quality. We started with a service log analysis of ad effectiveness with different articles placed together. The results showed that users exposed to the high-quality article consumed more ads than those exposed to the low-quality article. We then hypothesized that the exposure to high-quality articles leads to a high ad consumption rate. Motivated by this hypothesis, we conducted million-scale A/B testing to answer the following research questions:

- **RQ1:** To what extent does the quality of news articles affect ad consumption?
- **RQ2:** What kinds of users are likely to be affected by the news article quality in terms of ad consumption?
- **RQ3:** What kinds of ads are likely to be affected by the news article quality in terms of their consumption?

In the treatment group in A/B testing, we prioritized high-quality articles in the ranking while we did not explicitly prioritize high-quality articles in the control group. Our A/B testing revealed the following findings:

- **RQ1:** The number of clicks, conversions, and sales increased significantly in the treatment group.
- **RQ2:** Ad consumption increased regardless of user action volume. Notably, ad consumption of information seekers increased, and there was a significant interaction between news quality and the behavioral intention to use the service.
- **RQ3:** The ad consumption increased for most ad genres. The ad consumption of financial ads increased significantly.

These insights into the relationship between news articles and ads will help to simultaneously optimize both news and ad effectiveness in rankings.

5.2 Preliminary

In this section, we describe a news service that we used for our study and define the quality of news articles.

5.2.1 Target Service

We analyzed user behavior on Gunosy[1], one of the most popular news applications in Japan. When a user visits the news service, he or she is given a ranking including news articles based on the his or her interest. At the same time, in-feed ads are presented in between news articles. When a user clicks on the title or image of an article in the ranking, the system displays the detailed content of the clicked article. Users can stop reading the content at any time, and, if stopped, they can return to the clicked position in the original ranking. In this way, a user can browse multiple articles in a ranking. The user can quit at any time during these series of behaviors.

The news platform we used collects user feedback including the scroll depth in an article, ad conversions[2], and clicks and impressions for both the articles and ads. Articles and ads are displayed to users in a ranking, and data are collected for each position in the ranking.

5.2.2 Article Quality

Definition

Based on the existing study [50], article quality is defined as a degree correlating to the following criteria.

Authenticity The content is authentic, has high credibility, and is not imaginary or exaggerated [81].

Expression The content is objective, accurate, and the information volume is adequate but not redundant. In addition, the content is not vulgar, violent, or bloody, and does not contain pornographic words.

Headline The headline information is consistent with the body content and is not fake, exaggerated, or vulgar.

[1]https://gunosy.com

[2]*Conversion* refers to the user behavior of purchasing products, installing other services introduced in the advertisement, or requesting detailed information about the advertised product.

In this paper, we define high-quality articles as those satisfying all of the criteria, whereas low-quality articles are those violating at least one of the criteria. We note that our definition of low-quality articles is related to the concept of *clickbait* [8, 19].

Quality Judgement

We judged the quality of news articles by using experts and a rule-based method. News articles to be judged were selected from six popular news categories: social, economic, entertainment, technology, sport, and cooking. First, we developed a list of low-quality words, such as vulgar, violent, or cruel words. Then, we retrieved articles containing one or more low-quality words in their headline or body, and judged them as low-quality. Experts were employed to find high-quality articles because it is difficult to judge the objectivity and accuracy of the content and appropriate length of articles by simply using heuristics [50]. Three experts were involved in identifying a high-quality article. An article is considered high-quality if it is judged by all experts to meet the quality criteria.

These experts were members of the content quality team working for the news platform. The content quality team is comprised of professionals involved in news recommendations from a qualitative perspective. For example, they are responsible for preventing the diffusion of news articles that could be harmful.

5.2.3 Metrics

We used the following metrics to measure the consumption of news articles and ads:

Click-Through Rate (CTR) This metric is defined as the number of clicks divided by the number of impressions. CTR is used for both article and ad analysis. CTR is a metric of the likelihood that users will click on the article or ad.

Article Scroll Rate This metric is defined as the length of the article read by the user divided by the total length of the article. If this metric is low, it may indicate that users are not likely to have read the article until the end.

Conversion Rate (CVR) This metric is defined as the number of conversions divided by the number of clicks. When this metric is low, it suggests that many users clicked on the ad but did not purchase it.

Sales/Users This metric is defined as the sales divided by the number of unique users. The sales come from advertisements when a user clicks.

Since there is a business risk to exposing raw metrics, all values are normalized to $[0, 1]$, but their relative differences are shown in the Figure 5.1, Figure 5.2 and Figure 5.3.

5.3 Log Analysis

In this section, we investigate the effect of of news article quality on ad consumption through service log analysis[3].

5.3.1 Data

We collected 30 days of user feedback on a million-scale news service described in Section 5.2.1. During this period, the number of high-quality articles was 537 and the number of low-quality articles was 4,472. The total number of user sessions[4] in the collected logs was more than 10 million. To reduce bias, such as different amounts of user activity and position bias, we only analyzed the user behaviors at *top section* of the ranking, which consists of three articles at the first, second and third position and one advertisement at the fourth position in the ranking.

5.3.2 Article Consumption

We first investigated the effect of article quality on the article consumption. Figure 5.1 shows the relationship between the article quality and article consumption measured by the article CTR and scroll rate. The figure is a box-and-whisker plot of the article CTR and scroll rate grouped by the article quality. The median

[3]All values are normalized to $[0, 1]$, and their relative differences are shown in the Figure 5.1, Figure 5.2 and Figure 5.3.

[4]Session means the time interval between when a user opens and closes the mobile news service.

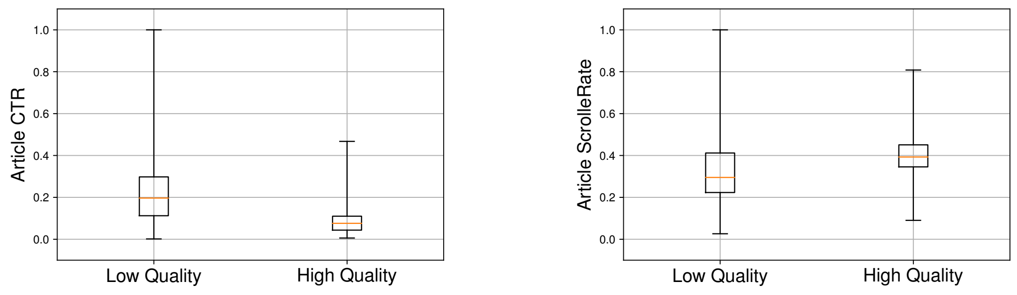

Figure 5.1: The Quality Effect on Article Consumption.

article CTR for low-quality articles is higher than the median article CTR for high-quality articles ($p < 0.05$, chi-square test for the average value), and the median article scroll rate for low-quality articles is lower than that for high-quality articles ($p < 0.05$, unpaired t-test for the average value).

These results suggest that low-quality articles are more likely to be clicked, but not be read to the end compared to high-quality articles. It is consistent with previous studies [50] that the result of article CTR and scroll rate vary for different levels of article quality.

5.3.3 Ad Consumption

We then investigate the effect of article quality on the ad consumption. Figure 5.2 shows the ad CTR and CVR for two conditions in which the ad was exposed together with a high- or low-quality article in the ranking. Note that, if both high- and low-quality articles exist in the ranking, the denominator of metrics for both high- and low-quality (i.e., the number of impressions or clicks) was incremented. The left figure in Figure 5.2 shows the ad CTR, while the right figure shows the ad CVR. When an advertisement is exposed with high-quality articles, the ad CTR and CVR are higher than those with low-quality articles ($p < 0.05$, chi-square test for each of ad CTR and CVR).

The result of the ad CTR may imply that low-quality articles got more users' attention than the advertisement, which led to the ad CTR being compromised. Another possible explanation for a high CVR is that users are more likely to trust the service with a high-quality article, which results in deeper engagement in the service (i.e., conversion).

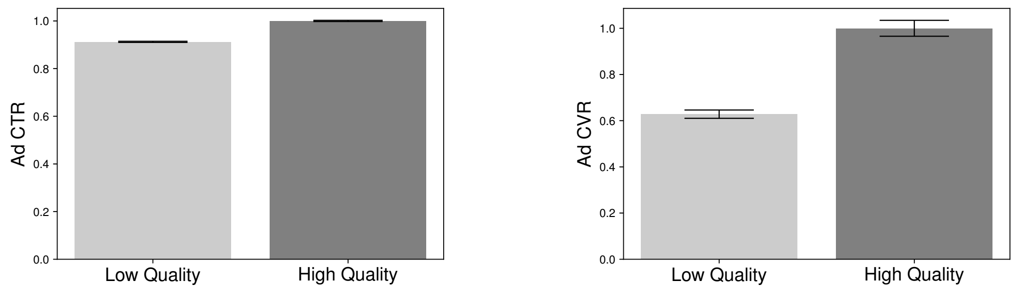

Figure 5.2: The Quality Effect on Ad Consumption. Error bar shows the 95% confidence interval for the Ad CTR and CVR.

5.4 Online Experiment

From the results of our log analysis, we hypothesize that displaying high-quality articles can lead to higher ad consumption. In this section, we conduct million-scale A/B testing to answer the following research questions:

- **RQ1:** To what extent does the quality of news articles affect ad consumption?
- **RQ2:** What kinds of users are likely to be affected by the news article quality in terms of ad consumption?
- **RQ3:** What kinds of ads are likely to be affected by the news article quality in terms of their consumption?

5.4.1 Setting

As in the log analysis, we conducted A/B testing on online news services described in Section 5.2.1. The duration of the A/B test was one month, which did not overlap with the log analysis period. In this A/B testing, we only focused on the effect of high-quality articles. We were not allowed to expose low-quality articles intentionally, since it can sacrifice user experience and ad effectiveness as the log analysis suggested.

The A/B testing target users in the treatment and control groups were randomly assigned using a 1:1 ratio from a subset of all users. High-quality articles were preferentially presented for users in the treatment group, more specifically,

Table 5.1: The Overall A/B Testing Result. Ad consumption was increased significantly in the treatment group for both iOS and Android users.

	AdCTR	AdCVR	Sales/Users	ArticleCTR
iOS User	+1.71%	+2.37%	+3.61%	-3.19%
Android User	+3.17%	+4.27%	+5.59%	-4.13%

they were inserted into the *top section*, or the top three positions in the ranking. On the other hand, the ranking generated from the existing recommendation algorithm was displayed in the control group. High-quality articles were those used in the log analysis. High-quality articles presented to users were rotated at least once every five hours between 7:00 a.m. and 10:00 p.m.

5.4.2 Results

RQ1: To what extent does the quality of articles affect advertisement consumption?

Table 5.1 shows the overall results of A/B testing. The results are divided into iOS users and Android users because we know from past experiments that results differ depending on the OS type.

As an overall result, all advertising metrics such as CTR, CVR, and Sales/Users improved significantly in the treatment group ($p < 0.05$ in chi-square test for AdCTR and AdCVR, and $p < 0.05$ in unpaired t-test for Sales/Users). This result indicates that high-quality articles increased the consumption of ads. We note that the article CTR was decreased in the treatment group ($p < 0.05$, chi-square test). As can also be seen in the log analysis, high-quality articles received fewer clicks than low-quality articles.

To answer **RQ1:** AdCTR, AdCVR, and Sales/Users improved significantly in the treatment group in which high-quality articles were presented at the top section of the ranking.

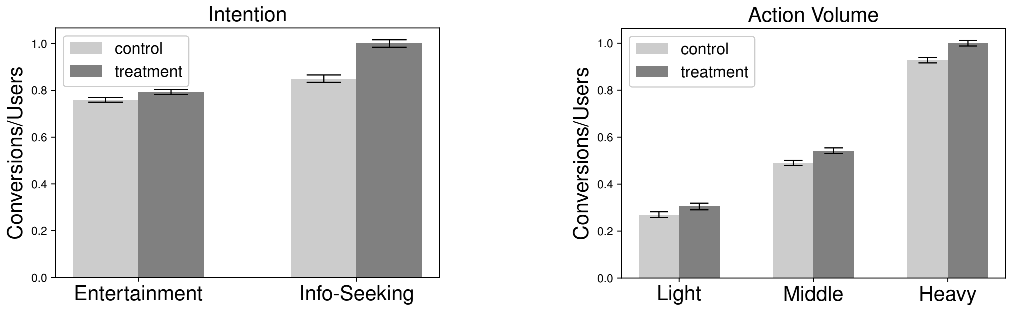

Figure 5.3: The Ad Consumption by User Segment.

RQ2: What kinds of users are likely to be affected by the news article quality in terms of ad consumption?

To investigate in detail the effects of article quality, we compare the results by user segments. We segmented users by two factors: intention to use the news service and action volume.

We divided the users into two groups by their intention: information-seeking and entertainment. Inspired by previous work [5], we define information-seekers as those whose clicks for economic or social topics account for more than 20% of their clicks in the past month, and users with entertainment intention as the others. We set the threshold at 20% to balance the sample size of the both groups.

Action volume is measured by the number of user clicks in the past month, and is divided into three groups: light, middle, and heavy. We used the 33.3% and 66.6% percentiles for the number of clicks to divide users by the action volume. Users in the 33% percentile or lower were considered light users whereas users in the 66% percentile or above were considered heavy users. Middle users fell in between.

Figure 5.3 shows the differences of the number of conversions grouped by the user segments, in which the left figure shows those by the user intention and right figure shows those by the action volume. We can see that information seekers in the treatment group were more influenced than users with entertainment intent in the treatment group. By action volume, all type of users had more ad consumption in the treatment group.

We conducted regression analysis by using a generalized linear model (GLM) [55] with a binomial distribution and a logit link function. To see the interaction be-

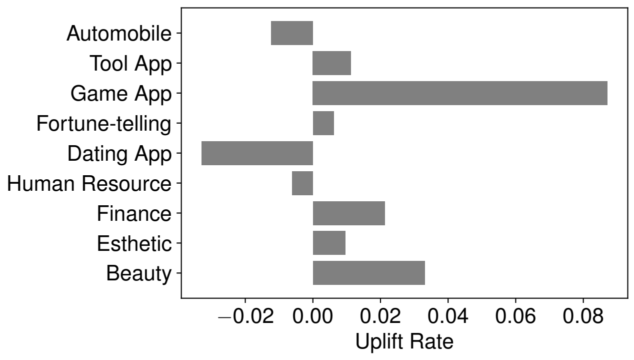

Figure 5.4: The Ad Consumption by Ad Genre.

tween the A/B testing group and both intention and action volume, we formulated the regression model to predict conversion as follows: conversion $\sim$ group + intention + action volume + group $\times$ intention + group $\times$ action volume. The regression result shows that the interaction term of the A/B testing group and the intention was significant ($p < 0.05$, Wald test) while the interaction term of the A/B testing group and the action volume was not significant.

Interestingly, information seekers were more likely to accept ads. This may imply that information seekers had more attention toward ads because of the high-quality articles placed above the ads, which contained objective and accurate information.

To answer **RQ2:** Ad consumption increased regardless of the user intention or user action volume. Especially, ad consumption of information seekers increased, and there existed an interaction between the news quality and user intention.

RQ3: What kinds of ads are likely to be affected by the news article quality in terms of their consumption?

Figure 5.4 shows the uplift rate of conversions per ad genre. We can see that the ad consumption was boosted by high-quality articles for most of the ad genres except for *Dating App* and *Automobile*. A chi-square test with Holm correction revealed that the *Finance* genre was the only genre that was statistically significant ($p < 0.05$), and this was beneficial to our news platform. Because the ad consumption for Finance genre was less than the other news services in the country where the news platform is located, increasing the ad consumption for the Finance genre was challenging for our news platform.

To answer **RQ3:** Most of the genres showed increase in ad consumption, among which the increase for the finance genre was significant.

5.5 Summary

In this study, we investigated the impact of article quality on ad consumption in news services. To investigate the effect, we first conducted a log-based analysis. Ad consumption was better for high-quality articles compared to low-quality articles. To verify in detail, we conducted a million-scale online experiment using A/B testing. The A/B test results showed that ad CTR, CVR, and sales improved significantly in the treatment group that explicitly involved high-quality articles. We also found that the ad consumption of users who prefer social or economic topics increased. These insights regarding news and advertisements will help optimize news and ad effectiveness in rankings.

Chapter 6

Conclusions

6.1 Summary

This thesis discussed algorithms and evaluations to address the challenges of news recommender systems. Three research topics addressed in this thesis are summarized as follows:

An Accurate and Efficient Embedding Method for News Recommendations

We proposed two techniques for accurate and efficient hyperbolic embedding for real-world recommender systems. The first technique is regularization. We found that the graphs of various recommendation datasets exhibit hierarchical or tree-like structures suitable for hyperbolic embeddings, while these structures are not well modeled by the original hyperbolic embedding approaches. Therefore, we introduce a regularization term in the objective function of the hyperbolic embedding for forcibly reflecting hierarchical or tree-like structures. The second technique is an efficient embedding method, which only updates the embedding of items that are recently added in a recommender system. In an offline evaluation with various recommendation datasets, we found that the regularization enforcing hierarchical or tree-like structures improved HR@10 by as much as +9% compared to hyperbolic embedding without regularization. Moreover, the evaluation result showed that our model update technique could achieve not only greater efficiency but also more robustness. Finally, we applied our proposed techniques to a million-scale news recommendation service and conducted an A/B test, which demonstrated that even *10-dimension* hyperbolic embedding successfully increased the number of clicks by +3.7% and dwell time by +10%.

An Interleaving Method for Evaluating Post-click Metrics

We proposed an efficient method for comparing the post-click metrics (e.g., dwell time and conversion rate) of multiple rankings in online experiments. The proposed method involves (1) the decomposition of the post-click metric measurement of a ranking into a click model estimation and a post-click metric measurement of each item in the ranking, and (2) interleaving of multiple rankings to produce a single ranking that preferentially exposes items possessing a high population variance. The decomposition of the post-click metric measurement enables the free layout of items in a ranking and focuses on the measurement of the post-click metric of each item in the multiple rankings. The interleaving of multiple rankings reduces the sample variance of the items possessing a high population variance by optimizing a ranking to be presented to the users so that those items received more samples of the post-click metric. In addition, we provide a proof that the proposed method leads to minimization of the evaluation error in the ranking comparison and propose two practical techniques to stabilize the online experiment. We performed a comprehensive simulation experiment and a real service setting experiment. The experimental results revealed that (1) the proposed method outperformed existing methods in terms of efficiency and accuracy, and the performance was especially remarkable when the input rankings shared many items, and (2) the two stabilization techniques successfully improved the evaluation accuracy and efficiency.

Evaluating the Effects of News Articles on Ad Consumption

We investigated the effect of news articles on users' ad consumption and shows the dependency between news and ad effectiveness. We conducted a service log analysis and showed that sessions with high-quality news article exposure had more ad consumption than those with low-quality news article exposure. Based on this result, we hypothesized that exposure to high-quality articles will lead to a high ad consumption rate. Thus, we conducted million-scale A/B testing to investigate the effect of high-quality articles on ad consumption, in which we prioritized high-quality articles in the ranking for the treatment group. The A/B test showed that the treatment group's ad consumption, such as the number of clicks, conversions, and sales, increased significantly while the number of article clicks decreased. We also found that users who prefer a social or economic topic had more ad consumption by stratified analysis. These insights regarding news articles and advertisements will help optimize news and ad effectiveness in rankings considering their mutual influence.

Finally, technical and social contributions of this research are summarized as follows:

6.1.1 Technical Contributions

- Developing an accurate and efficient method to recommend news utilizing the hyperbolic space.
- Developing an efficient method to evaluate post-click metrics.
- Giving insights between advertisement and news in the ranking of recommender systems.

6.1.2 Social Contributions

- Enabling users to access news in a timely way utilizing their preferences.
- Enabling researchers and companies to evaluate user satisfaction without compromising user experience compared to the traditional setting.
- Enhancing the news provider's profit from advertisements while maintaining the quality of news recommendations.

6.2 Future Directions

There are still several research questions to be addressed in the future. In this study, we first focused on collaborative-filtering-based hyperbolic embedding employing user interactions. In the future, we plan to build the content-based and a collaborative-filtering-based hybrid recommender system using compact hyperbolic embedding. Another possible direction is to integrate pre-trained language models and hyperbolic embedding models for zero-shot learning to deal with the cold-start problem. Furthermore, we focused on building an online evaluating method. We plan to extend our framework to include a bandit problem in post-click optimization. Using the post-click bandit algorithm, we can optimize dwell time and ad effectiveness in the news application. We might need to employ more sophisticated user click models to extend our framework because the bias issue would be critical in other problem settings. From the aspect of sponsored ranking analysis, we plan to conduct a study to investigate how advertisement affects the user's subjective perception of news ranking. Finally, we will utilize the insight of the sponsored ranking to enhance user satisfaction and revenue in real-world news applications.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Tetsuji Sato, Assoc. Prof. Makoto P. Kato and Assoc. Prof. Hai-Tao Yu for the continuous support of my Ph.D. study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Nobutaka Suzuki and Prof. Toshiyuki Amagasa for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

Finally, my sincere thanks go to all of the members in Gunosy Inc., especially Yoshifumi Seki of Gunosy, who provided me an opportunity to join the Ph.D. program while I am working at Gunosy. They have a deep understanding of the research topics and gave me a lot of time to do the research. Without their precious support, it would not be possible to conduct this research.

Bibliography

[1] Media guide, 2022. http://static.smartnews.com/smartnews-ads/key_mediaguide_smartnews.pdf.

[2] M. Abu-Ata and F. F. Dragan. Metric tree-like structures in real-world networks: An empirical study. *Netw.*, 67(1):49–68, 2016.

[3] A. O. Alanazi, M. Sanderson, Z. Bao, and J. Kim. The impact of ad quality and position on mobile serps. In *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval*, pages 318–322, 2020.

[4] J. Allan, B. Carterette, J. A. Aslam, V. Pavlu, B. Dachev, and E. Kanoulas. Million query track 2007 overview. Technical report, University of Massachusetts Amherst, 2007.

[5] M. A. Amazeen. News in an era of content confusion: Effects of news use motivations and context on native advertising and digital news perceptions. *Journalism & Mass Communication Quarterly*, 97(1):161–187, 2020.

[6] A.-L. Barabási and E. Bonabeau. Scale-free networks. *Scientific American*, 288(5):60–69, 2003.

[7] O. Barkan and N. Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[8] J. N. Blom and K. R. Hansen. Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics*, 76:87–100, 2015.

[9] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[10] B. Brost, I. J. Cox, Y. Seldin, and C. Lioma. An improved multileaving algorithm for online ranker evaluation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 745–748, 2016.

[11] G. Burtini, J. Loeppky, and R. Lawrence. A survey of online experiment design with the stochastic multi-armed bandit. *arXiv preprint arXiv:1510.00757*, 2015.

[12] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier. Word2vec applied to recommendation: Hyperparameters matter. In *Proceedings of the 2018 ACM Conference on Recommender Systems*, pages 352–356, 2018.

[13] B. P. Chamberlain, S. R. Hardwick, D. R. Wardrope, F. Dzogang, F. Daolio, and S. Vargas. Scalable hyperbolic recommender systems. *arXiv:1902.08648*, 2019.

[14] I. Chami, Z. Ying, C. Ré, and J. Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 4869–4880, 2019.

[15] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Trans. Inf. Syst.*, 30(1):1–41, 2012.

[16] D. Chen, S. L. Sain, and K. Guo. Data mining for the online retail industry: A case study of rfm model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, 19(3):197–208, 2012.

[17] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, pages 456–464, 2019.

[18] W. Chen, W. Fang, G. Hu, and M. W. Mahoney. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics*, 9(4):434–491, 2013.

[19] Y. Chen, N. J. Conroy, and V. L. Rubin. Misleading online content: recognizing clickbait as "false news". In *Proceedings of the 2015 ACM on workshop on multimodal deception detection*, pages 15–19, 2015.

[20] A. Chuklin, I. Markov, and M. De Rijke. *Click models for web search*. Morgan & Claypool Publishers, 2015.

[21] C. L. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2009 web track. Technical report, University of Waterloo, 2009.

[22] H. Cramer. Effects of ad quality & content-relevance on perceived content quality. In *proceedings of the 33rd annual ACM conference on human factors in computing systems*, pages 2231–2234, 2015.

[23] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 1st ACM International Conference on Web Search and Data Mining*, pages 87–94, 2008.

[24] M. F. Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 2019 ACM Conference on Recommender Systems*, pages 101–109, 2019.

[25] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296, 2010.

[26] A. Deng, Y. Xu, R. Kohavi, and T. Walker. Improving the sensitivity of online controlled experiments by utilizing pre-experiment data. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 123–132, 2013.

[27] G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 331–338, 2008.

[28] M. Farajtabar, Y. Chow, and M. Ghavamzadeh. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pages 1447–1456. PMLR, 2018.

[29] S. Feng, L. V. Tran, G. Cong, L. Chen, J. Li, and F. Li. Hme: A hyperbolic metric embedding approach for next-poi recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1429–1438, 2020.

[30] J. Garcia-Gathright, B. St. Thomas, C. Hosey, Z. Nazari, and F. Diaz. Understanding and evaluating user satisfaction with music discovery. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64, 2018.

[31] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé. Offline a/b testing for recommender systems. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 198–206, 2018.

[32] M. Grbovic and H. Cheng. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 311–320, 2018.

[33] A. Gruson, P. Chandar, C. Charbuillet, J. McInerney, S. Hansen, D. Tardieu, and B. Carterette. Offline evaluation to make decisions about playlistrecommendation algorithms. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, pages 420–428, 2019.

[34] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 2020 International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 639–648, 2020.

[35] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 2017 International Conference on World Wide Web*, pages 173–182, 2017.

[36] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 2016 International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 549–558, 2016.

[37] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *arXiv:1511.06939*, 2015.

[38] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[39] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu. Tencentrec: Real-time stream recommendation in practice. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 227–238, 2015.

[40] J.-Y. Jiang, P. H. Chen, C.-J. Hsieh, and W. Wang. Clustering and constructing user coresets to accelerate large-scale top-k recommender systems. In *Proceedings of The Web Conference 2020*, WWW '20, pages 2177–2187. Association for Computing Machinery, 2020.

[41] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *ACM SIGIR Forum*, volume 51, pages 4–11. Acm New York, NY, USA, 2017.

[42] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2):7–es, 2007.

[43] T. Joachims, A. Swaminathan, and T. Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 781–789, 2017.

[44] A. Katal, M. Wazid, and R. H. Goudar. Big data: issues, challenges, tools and good practices. In *2013 Sixth international conference on contemporary computing (IC3)*, pages 404–409. IEEE, 2013.

[45] K. Klemm and V. M. Eguiluz. Highly clustered scale-free networks. *Physical Review E*, 65(3):036123, 2002.

[46] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann. Online controlled experiments at large scale. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1168–1176, 2013.

[47] P. Kolyvakis, A. Kalousis, and D. Kiritsis. Hyperbolic knowledge graph embeddings for knowledge base completion. *arXiv:1908.04895*, 2019.

[48] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 2016 ACM Conference on Recommender Systems*, pages 59–66, 2016.

[49] J. Lin, S. Mohammed, R. Sequiera, and L. Tan. Update delivery mechanisms for prospective information needs: An analysis of attention in mobile users. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 785–794, 2018.

[50] H. Lu, M. Zhang, W. Ma, Y. Shao, Y. Liu, and S. Ma. Quality effects on user preferences and behaviorsin mobile news streaming. In *The World Wide Web Conference*, WWW '19, pages 1187–1197, 2019.

[51] H. Lu, M. Zhang, W. Ma, C. Wang, F. xia, Y. Liu, L. Lin, and S. Ma. Effects of user negative experience in mobile news streaming. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 705–714, 2019.

[52] I. MacKenzie, C. Meyer, and S. Noble. How retailers can keep up with consumers. *McKinsey & Company*, 18:1, 2013.

[53] J. Mao, Y. Liu, K. Zhou, J.-Y. Nie, J. Song, M. Zhang, S. Ma, J. Sun, and H. Luo. When does relevance mean usefulness and user satisfaction in web search? In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 463–472, 2016.

[54] P. Melville and V. Sindhwani. Recommender systems. *Encyclopedia of machine learning*, 1:829–838, 2010.

[55] J. A. Nelder and R. W. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[56] J. Ni, J. Li, and J. McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 188–197. Association for Computational Linguistics, Nov. 2019.

[57] M. Nickel and D. Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, pages 6338–6347, 2017.

[58] M. Nickel and D. Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018.

[59] S. Okura, Y. Tagami, S. Ono, and A. Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 2017 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1933–1942, 2017.

[60] S. Okura, Y. Tagami, S. Ono, and A. Tajima. Embedding-based news recommendation for millions of users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1933–1942, 2017.

[61] H. Oosterhuis and M. de Rijke. Sensitive and scalable online evaluation with theoretical guarantees. In *Proceedings of the 26th ACM International on Conference on Information and Knowledge Management*, pages 77–86, 2017.

[62] H. Oosterhuis and M. de Rijke. Taking the counterfactual online: Efficient and unbiased online evaluation for ranking. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, pages 137–144, 2020.

[63] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE, 2008.

[64] A. Poyarkov, A. Drutsa, A. Khalyavin, G. Gusev, and P. Serdyukov. Boosted decision tree regression adjustment for variance reduction in online controlled experiments. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 235–244, 2016.

[65] T. Qin, T.-Y. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. August 2016.

[66] F. Radlinski and N. Craswell. Optimized interleaving for online retrieval evaluation. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*, pages 245–254, 2013.

[67] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM International Conference on Information and Knowledge Management*, pages 43–52, 2008.

[68] E. Ravasz and A.-L. Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.

[69] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 251–258, 2008.

[70] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[71] Y. Saito. Doubly robust estimator for ranking metrics with post-click conversions. In *Proceedings of the 14th International ACM Conference on Recommender Systems*, pages 92–100, 2020.

[72] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, 2002.

[73] A. Schuth, R.-J. Bruintjes, F. Buüttner, J. van Doorn, C. Groenland, H. Oosterhuis, C.-N. Tran, B. Veeling, J. van der Velde, R. Wechsler, et al. Probabilistic multileave for online retrieval evaluation. In *Proceedings of the 38th international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 955–958, 2015.

[74] A. Schuth, K. Hofmann, and F. Radlinski. Predicting search satisfaction metrics with interleaved comparisons. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 463–472, 2015.

[75] A. Schuth, F. Sietsma, S. Whiteson, D. Lefortier, and M. de Rijke. Multileaved comparisons for fast online evaluation. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 71–80, 2014.

[76] R. Shimizu, Y. Mukuta, and T. Harada. Hyperbolic neural networks++. *arXiv preprint arXiv:2006.08210*, 2020.

[77] Y. Su, L. Wang, M. Santacatterina, and T. Joachims. Cab: Continuous adaptive blending for policy evaluation and learning. In *International Conference on Machine Learning*, pages 6005–6014, 2019.

[78] Y. Tay, L. A. Tuan, and S. C. Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 583–591, 2018.

[79] L. Vinh Tran, Y. Tay, S. Zhang, G. Cong, and X. Li. Hyperml: A boosting metric learning approach in hyperbolic space for recommender systems. In *Proceedings of the 2020 International Conference on Web Search and Data Mining*, pages 609–617, 2020.

[80] E. M. Voorhees and D. Harman. Overview of trec 2003. In *TREC*, pages 1–13, 2003.

[81] S. Vosoughi, D. Roy, and S. Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.

[82] C. Wang, Y. Liu, M. Zhang, S. Ma, M. Zheng, J. Qian, and K. Zhang. Incorporating vertical results into search click models. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 503–512, 2013.

[83] W. Wang, J. Jin, J. Hao, C. Chen, C. Yu, W. Zhang, J. Wang, X. Hao, Y. Wang, H. Li, J. Xu, and K. Gai. Learning adaptive display exposure for real-time advertising. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 2595–2603, 2019.

[84] X. Wang, N. Golbandi, M. Bendersky, D. Metzler, and M. Najork. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 610–618, 2018.

[85] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[86] Y. Wu and C. Zhang. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics*, page P43, 2011.

[87] H. Xie and J. Aurisset. Improving the sensitivity of online controlled experiments: Case studies at netflix. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 645–654, 2016.

[88] J. Yan, Z. Xu, B. Tiwana, and S. Chatterjee. Ads allocation in feed via constrained optimization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, pages 3386–3394, 2020.

[89] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan. Beyond clicks: Dwell time for personalization. In *Proceedings of the 8th International ACM Conference on Recommender Systems*, pages 113–120, 2014.

[90] Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *Proceedings of the 19th international conference on World wide web*, pages 1011–1018, 2010.

[91] X. Zhao, X. Zheng, X. Yang, X. Liu, and J. Tang. Jointly learning to recommend and advertise. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, pages 3319–3327, 2020.

Publications

Journal Articles

1. 飯塚洸二郎，加藤誠，関喜史,” ニュース記事の品質が広告消費行動に与える影響の調査”, 日本データベース学会和文論文誌, Vol.20-J, No.14, 2022, 7 pages.

2. Kojiro Iizuka, Makoto P. Kato and Yoshifumi Seki, ”Dynamic Hyperbolic Embeddings with Graph-Centralized Regularization for Recommender Systems”, Journal of Information Processing, Vol.29, 2021, pp.725-734.

Conference Papers

1. Kojiro Iizuka, Yoshifumi Seki and Makoto P. Kato, ”Decomposition and Interleaving for Variance Reduction of Post-click Metrics”, Proc. of the 7th ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR 2021), July 2021, pp. 221-230.

2. Kojiro Iizuka, Yoshifumi Seki and Makoto P. Kato, ”The Effect of News Article Quality on Ad Consumption”, Proc. of the 30th ACM International Conference on Information and Knowledge Management (CIKM 2021), October 2021, pp. 3107-3111.

3. Kojiro Iizuka, Takeshi Yoneda and Yoshifumi Seki, ”Greedy Optimized Multileaving for Personalization”, Proc. of the 13th ACM International Conference on Recommender Systems (RecSys 2019), September 2019, pp. 413-417.

Domestic Symposium

1. 飯塚洸二郎, 加藤誠” インターリービングにおける正確性と効率性の理論的考察”, 第 14 回データ工学と情報マネジメントに関するフォーラム (DEIM 2022), 2022 年.
2. 飯塚洸二郎, 加藤誠, 関喜史, ” ニュース記事の品質が広告消費行動に与える影響の調査”, 第 13 回データ工学と情報マネジメントに関するフォーラム (DEIM 2021), 2021 年.
3. 飯塚洸二郎, 関喜史, ” 双曲空間におけるニュース推薦のためのオンライングラフ埋め込み”, 第 12 回データ工学と情報マネジメントに関するフォーラム (DEIM 2020), 2020 年.
4. 飯塚洸二郎, 米田武, 関喜史, ” ニュースアプリケーションのパーソナライゼーションアルゴリズムに対するマルチリービング手法の比較”, 第 33 回人工知能学会全国大会 (JSAI 2019), 2019 年.

Appendix: Variance of $\overline{E}[x|r_i]$

Noting that $V[x+y] = V[x] + V[y]$ if x and y are independent, we obtain:

$$V[\overline{E}[x|r_i]] = V\left[\sum_{d\in r_i} \overline{P}(c_d = 1|r_i)\overline{E}[x|d]\right] = \sum_{d\in r_i} V\left[\overline{P}(c_d = 1|r_i)\overline{E}[x|d]\right]$$

Using $V[xy] = V[x]V[y] + E[x]^2 V[y] + V[x]E[y]^2$ (x and y are independent), the variance for each item is obtained as follows:

$$\begin{aligned} V\left[\overline{P}(c_d = 1|r_i)\overline{E}[x|d]\right] &= V[\overline{P}(c_d = 1|r_i)]V[\overline{E}[x|d]] \\ &+ E[\overline{P}(c_d = 1|r_i)]^2 V[\overline{E}[x|d]] \\ &+ E[\overline{E}[x|d]]^2 V[\overline{P}(c_d = 1|r_i)] \end{aligned}$$

Since $V[\overline{x}] = \sigma^2/n$ ($\overline{x}$ is the sample mean, σ^2 is the population variance, and n is the sample size), $V[\overline{P}(c_d = 1|r_i)]$ can be computed as $V[\overline{P}(c_d = 1|r_i)] = V[P(c_d = 1|d)]/n^i_d$, where n^i_d is the number of impressions of item d. We can also obtain $V[\overline{E}[x|d]] = V[x|d]/n^c_d$ where n^c_d is the number of clicks on item d.

Finally, we express the variance of $\overline{E}[x|r_i]$ as a function of the samples sizes n^i_d and n^c_d:

$$\begin{aligned} V[\overline{E}[x|r_i]] = \sum_{d\in r_i} \Bigg\{ &\frac{V[P(c_d = 1|d)]}{n^i_d}\frac{V[x|d]}{n^c_d} + E[\overline{P}(c_d = 1|r_i)]^2 \frac{V[x|d]}{n^c_d} \\ &+ E[\overline{E}[x|d]]^2 \frac{V[P(c_d = 1|d)]}{n^i_d} \Bigg\} := \sum_{d\in r_i} \phi_{d,r_i}(n^i_d, n^c_d) \end{aligned}$$

The function ϕ_{d,r_i} monotonically decreases for n^i_d and n^c_d when x is always positive. As $\overline{P}(c_d = 1|r_i)$ and $\overline{E}[x|d]$ are unbiased estimators, $E[\overline{P}(c_d = 1|r_i)]$ and $E[\overline{E}[x|d]]$ can be approximated by $\overline{P}(c_d = 1|r_i)$ and $\overline{E}[x|d]$ with a sufficient number of samples. Assuming that $P(a_d|d)$ follows a Bernoulli distribution, we can also approximate $V[P(c_d = 1|d)]$ by $\overline{P}(c_d = 1|r_i)(1 - \overline{P}(c_d = 1|r_i))$.