An Ensemble Approach for News Recommendation Based on Contextual Bandit Algorithms

Yu Liang

An Ensemble Approach for News Recommendation Based on Contextual Bandit Algorithms

by

Yu Liang

in partial fulfillment of the requirements for the degree of

Master of Science
in Computer Science

at the Delft University of Technology,
to be defended publicly on Friday September 29, 2017 at 3:00 PM.

Supervisor:	Prof. dr. Martha Larson	
Thesis committee:	Prof. dr. Martha Larson,	TU Delft
	Dr. Nava Tintarev,	TU Delft
	Dr. Frank Hopfgartner,	University of Glasgow

Abstract

News recommendation is a field different from traditional recommendation fields. News articles are created and deleted continuously with a very short life cycle. Users' preference is also hard to model since they can easily be attracted by things happening around them. With all those challenges, traditional recommendation approaches, such as content-based filtering and collaborative filtering, do not work well in the field. Simple recency-based or popularity-based recommenders do work well. However, even the recommender with the highest performance has its restriction. In this work, we build an ensemble model to combine the power of different recommenders. We build up a delegation model on top of several news recommenders based on various contextual bandit algorithms (a combination of multi-armed bandit algorithms and context information). The delegation model is responsible for delegating recommendation requests to the appropriate recommender with the purpose to maximize the Click Through Rate (CTR) from the recommendations and can update continuously with users' feedback. We evaluate the performance of our delegation-model-based recommender in both online and offline scenarios with the evaluation methods provided by CLEF-NEWSREEL Challenge 2017. Furthermore, we also evaluate the response time of our delegation model to see whether it is feasible to run online. The results show that our proposed delegation model can choose the appropriate recommender to serve the incoming requests each time, improve its performance regarding CTR and is feasible to run in real-world settings. Additionally, we also evaluate our delegation-model-based recommender with another evaluation metric, catalog coverage. In our future work, we would like to combine more recommenders and explore more context features to further improve CTR.

Keywords

News recommendation, Contextual bandit algorithms, Multi-armed bandit, Context, Recommender systems, Online evaluation, Offline evaluation, Ensemble recommender

Preface

The thesis paper is the final work of my two-year Master study in Computer Science at Delft University of Technology. This work is carried out on the Dutch national e-infrastructure with the support of SURF Cooperative.

The thesis is partially based on the CLEF NewsREEL News Recommendation Evaluation Lab, which benefits the whole thesis with a comprehensive dataset and two advanced evaluation methods. Two weeks before my thesis defense, I was lucky to have a chance to go to Trinity College Dublin for CLEF 2017 and present my paper on the CLEF NewsREEL Lab. It was also my very first time to attend an academic conference and turned out to be an amazing and unforgettable experience.

I would like to first express my gratitude to my supervisor, Prof. Martha Larson, who has inspired me a lot, for her professional guidance, encouragement and continuous support throughout the period of this work and making my attendance to the conference possible. I would also like to give my thanks to Babak Loni for his assistance and valuable advice during the whole project. Thanks Dr. Nava Tintarev and Dr. Frank Hopfgartner for being my thesis committee. Thanks Dr. Andreas Lommatzsch for his insights in the field. I would also like to thank all people from the RecSys meeting. Thanks for listening to my presentation and giving me useful comments and suggestions each time.

Additionally, I would like to thank one of my best friends Ruoqing Sun for accompanying me through the challenges of study and life in the past two years. I would also like to give my special thanks to my two cat friends, Pidgey and Choco, for accompanying me each time I worked late at night. It is great to have you in my life.

Last but not least, I would like to express my deepest thanks to my parents, who were always standing by my side and supporting me with their best wishes.

Yu Liang
Delft, September 2017

Contents

1 Introduction

1.1. Motivation

News recommendation is a field different from other recommendation fields. Things change too fast in the field. There are many challenges in the field. From the side of users, their news interests keep changing over time and can be easily influenced by what is happening around the world. Some previous work has shown that individual user's news interests can be influenced by regional news trends [1]. Users' preferences are generally hard to model in the field of news recommendation. Those news websites that require users to register and thus maintain an account for each user, can provide the personalized recommendation by tracking users' historical behaviors [2]. However, there are still many news websites that do not require users to sign up or log in. For those websites, although sometimes users can be tracked by cookies, it is even harder to capture their preference since the news websites almost know nothing about them except their possible temporal preference built based on their session behaviors. Meanwhile, the item set is more dynamic than the one in movie recommendation and music recommendation. News articles are created, updated and even deleted continuously depending what is happening around the world. Furthermore, timeliness is also an important factor in the field. Once popular news articles expire in a short time if people do not care about them anymore. All these challenges make news recommendation different from music recommendation and movie recommendation, where both users and items are much more static.

The result of the challenges is that traditional personalized recommendation methods, such as collaborative filtering, tend to fail in the field since users' profiles cannot be modeled. In addition, collaborative filtering methods, such as factorization machines, which work well in movie recommendation are not suitable for the field since the high computation complexity of the recommender would make it impossible to provide efficient recommendations within a time limit. Furthermore, content-based recommendation methods also fail since they will continue to recommend news articles with the similar content to users while users are more likely to be attracted by the popular news [3].

Current recommenders working well in the online news field are usually simple popularity-based recommenders and recommenders combined with temporal information, such as neighborhood-based collaborative filtering recommenders using temporal information proposed in the CLEF NewsREEL Challenge 2017 (see Sec.1.3 for details about CLEF NewsREEL Challenge) [4]. The commonalities of the recommenders are low computational complexity, which makes them able to provide efficient recommendations within a time limit, and the ability to keep up with the current news trends.

However, even the recommender with the highest performance has its restriction. It is a wise decision to build an ensemble model to combine the power of different recommenders. Based on this idea, previous work in the CLEF NewsREEL Challenge combined several recommender [5–8] using a delegation model. Each time when there is a news recommendation request, the delegation model is responsible for delegating the request to the 'appropriate' news recommender. There are many ways to select the appropriate recommender: for example, it can be the one receiving most clicks in the most recent 15 minutes [5] or the one with the best performance under the specific context. However, those delegation models are not well built. Some of them fail to consider the context information which can be helpful for recommender choosing [6, 7], and none of them are able to update automatically in the online scenario with the users' feedback.

Based on these on challenges and previous work, we ask the question of whether we can build a delegation model that is able to choose an appropriate recommender for recommendation requests given the context and meanwhile also able to update continuously with users' feedback. In the next section, we will discuss our research questions in detail.

1.2. Scope and Research Questions

This thesis addresses the problem of news recommendation, as represented by the CLEF NewsREEL Challenge, explained further in Sec. 1.3. Previous work in the CLEF NewsREEL Challenge has shown that context can help to select the best-suited recommender from a recommender algorithm ensemble [6, 7]. With our work, we ask the question of whether the selection strategy can be automatically and continuously updated from the incoming requests and their corresponding feedback (click triggered by users). By studying the related work, we found a Multi-armed Bandit algorithm is a good way to serve the purpose. Our main research question is:

- Is there a way in which the delegation strategy is able to update continuously based on users' feedback so that it can choose the appropriate recommender to serve recommendations during the online news recommendation?

  (Can a Multi-armed Bandit algorithm be a way to continuously update the delegation strategy from the click behaviors learned from users during the online news recommendation?)

This research question is addressed by answering several sub-questions:

- Can a Multi-armed Bandit algorithm combined with context help to build a better delegation strategy in the online news recommendation? Furthermore, what kind of context can be exploited and how the context is combined?

- Is it possible for the proposed delegation model to run online in order to serve the recommendation requests in the online news recommendation scenario? In other words, can the delegation model feasible to run online to provide valid recommendation requests within a specified time limit?

Here, context is a set of feature vectors which are encoded in the recommendation requests and contain information about the current user and the news articles he/she is currently reading, such as user identification by cookies and from which publisher the recommendation requests come from.

1.3. The CLEF NewsREEL Challenge

The CLEF NewsREEL Challenge [9, 10] is a challenge in which participants are asked to provide effective real-time news recommendation. The challenge provides participants with a platform to evaluate their recommender algorithms with millions of real-world users. We took part in both tasks [11]: Task 1 (NewsREEL Live) [12] and Task 2 (NewsREEL Replay) [13]. In the Live task, participants are required to generated real-time news recommendations for users in an online scenario. The Replay task, on the other hand, reproduces the online recommendation scenario. In this task, a simulator is implemented to simulate online data stream by replaying the log data recorded from the online scenario. This task provides participants a way to evaluate their news recommenders in a near-online scenario. In the thesis, we also evaluate our recommender both online and offline (near-online) with the two evaluation scenarios provided by the NewsREEL Challenge.

1.4. Contributions

The contribution of our thesis is twofold. Firstly, to the best of our knowledge, we are the first to combine contextual bandit algorithm with recommender algorithms in the news recommendation. Although contextual bandit algorithm has been used in the news recommendation field before [14], it has been directly combined with news articles and served as a recommender to decide which news article to recommend each time instead of served as a delegation model. The contextual bandit algorithm in our work deals with more complex objectives, different recommenders, rather than simple news articles.

Specifically, in our work, a delegation model based on the contextual bandit algorithm is built to choose appropriate recommender for each recommendation request to improve the recommendation performance. The delegation model is able to continuously update itself with users' feedback in the online news recommendation scenario. The delegation model is also scalable, making it easier to combine more context and recommender algorithm in the future. It is also feasible to run the model in the online scenario and the model is able to provide valid recommendations within a time limit. Secondly, another evaluation metric, catalog coverage, is tried to evaluate the system performance in addition to the commonly used metric CTR. Although CTR is important from the company side, we argue it should not be the only metric from the user side and from a company side for a long run. Catalog coverage is also important from the user side since it is rather meaningless to recommend popular news articles to users continuously in order to reach a high CTR but ignore users' possible interests in the news articles from the long tail. Meanwhile, satisfying users' needs would benefit the company's development in a long run.

1.5. Thesis Structure

The thesis is structured as follows:

- **Chapter 2 Background and Related Work**:

  In this chapter, we discuss the background of the recommender system, the characteristics of news recommendation, the definitions as well as algorithms involved with multi-armed bandit problems and evaluation metric for the online news recommendation.

- **Chapter 3 - Delegation Model**:

  In this chapter, we describe our delegation model, including the recommender algorithms used, the working flow of our delegation model and how context is exploited in the delegation model.

- **Chapter 4 - Implementation**:

  In this chapter, we provide the details and practical implementation of our online news recommender.

- **Chapter 5 - Evaluation**:

  In this chapter, we present how the experiments are conducted based on the online and near-online evaluation framework provided by the NewsREEL challenge.

- **Chapter 6 - Discussion and Future Work**

  In this chapter, we discuss the evaluation results from the previous chapter, draw conclusions from the whole thesis, and, last but not least, discuss possible future work.

2 Background and Related Work

In this chapter, we introduce the background and related work of the whole project with respect to the three most important aspects: recommender systems, multi-armed bandit algorithms and evaluation metrics. When talking about recommender systems, we put our focus on news recommendation, a field different from common recommendation fields, e.g., music recommendation, where users and items are much more static. As the whole delegation model is based on multi-armed bandit algorithms, several multi-armed bandit algorithms and contextual bandit algorithms (multi-armed bandit algorithm combined with context) are introduced in Sec. 2.2 and Sec. 2.3. In the last section, we describe the evaluation metric used to evaluate the performance of our news recommender, including Click Through Rate (CTR), recommendation response time and catalog coverage of the recommendation.

2.1. Recommender System

Over the past decades, recommender systems have become an effective information filtering technique to deal with the large amount of information across the Web. They have been implemented in various fields, including e-commerce (Amazon), music recommendation (Last.fm), movie recommendation (Netflix), Job Recommendation (Linkedin) and News Recommendation (Yahoo). Commonly used recommender methods can be classified into content-based filtering, collaborative filtering, hybrid method [15] and popularity-based recommendation methods [16].

2.1.1. Content-based Filtering

In the context-based filtering, users and items can both be represented by a set of attributes characterizing their profiles [17]. For example, in the movie recommendation field, the movie recommender can suggest movies to the user based on his/her preference about movie genres and directors. This method is good at tackling with the cold start problem that arises when new users or items come to the system since recommendations are always generated by matching the user and item profile. In addition, it is also easy to explain to users why a certain item is recommended. However, the downside of the method is it continuously recommends similar items to the user based on the user's profile and fails to generate any serendipitous recommendations. Furthermore, the content-based method requires the system to hold knowledge of both users' and items' profiles beforehand.

2.1.2. Collaborative Filtering

Another widely used method is collaborative filtering. Recommendations are given based on ratings or behaviors of other users [18]. The basic underlying idea is that other user's behaviors can be aggregated to predict a given user's preferences. For example, John is likely to prefer the movie liked by users who like the similar movies as John. Collaborative filtering method can further be classified as memory-based and model-based methods. Memory-based method compute users' preferences on items directly from the user-item matrix. Two widely used memory-based methods are user-user collaborative filtering and item-item collaborative filtering [18]. The basic idea behind user-user collaborative filtering is to find other users who have the similar behavior as the current user and use their ratings on the items to predict the current user's preference. While in item-item collaborative filtering, items that are similar to the items liked by the current user are recommended. In contrast, in model-based CF, a prediction model is trained from the user-item matrix. The prediction model is then used to generate recommendations for users. Collaborative filtering has already been implemented in various fields with good recommendation results, such as with Last.fm[1]. Besides, to provide a recommendation, it is not necessary for the system to know users' or items' attributes. However, it also has several limitations. Firstly, the U-I matrix built from real-world is very sparse and the sparsity of the U-I matrix makes it hard to provide reliable recommendations. Secondly, the number of both users and items grows continuously in practice. For this reason, a CF method would face a scaling problem. The CF approaches are extremely weak at solving the cold-start problems. It is hard for a CF approach to make a reliable recommendation to new users and new items will have little chance to be recommended. In some work, collaborative filtering is combined with content-based filtering to provide good recommendation results and at the same time to prevent cold-start problems [19].

2.1.3. Context-Aware Recommendation

In recent years, context-aware recommender systems have gained in popularity. Context-aware recommender systems, when providing recommendations, utilize additional information besides user and item information [20]. Previous work has shown contextual information used as additional information can help to better predict users' ratings on items [21]. For example, a user's ratings on a movie might depend on the situation, e.g., his/her current mood or the current location. Furthermore, it is now suggested in some application domains where recommender algorithms are driven by context [22]. Here, context is critical instead of additional. Studies have shown some domains are context-centric [22]. For example, in linear TV recommender systems, context can be more important than items [23]. It is suggested that users would be more likely to watch an interesting TV program in his/her preferred channel and time slots instead of a more interesting one in a different time slot. In the news recommendation field, context is also utilized to provide better recommendations [14]. We will discuss more details about news recommendation in the next section.

2.1.4. News Recommendation

In this section, we describe unique features of the news recommendation field.

- **Rapid Changing Item Set**

  Unlike most traditional recommender systems, like movie recommender system where item set are relatively static, news articles change fast and have a very short lifecycle [6]. The item set undergoes churn continuously [2]. In addition, once a news article reaches the end of its lifecycle (normally at most 3 days), it will be considered as outdated and not be visited anymore. The short life-cycle of news articles would cause a continuous cold-start problem from the item side.

- **Unregistered Users**

  Some work in the news recommendation field is able to provide personalized recommendation to users based on users' click behaviors [1, 2] making use of various collaborative filtering methods. However, in some news portals, users are not required to register and log in [24]. In the NewsREEL Challenge, registration is also not required and users can be tracked by their cookies, only if they allow the use of cookies.

- **Weather Problem**

  Users' interests change fast in the news field and are always affected by the popular events currently happening in the world. News articles related to those popular events will suddenly gain its popularity, e.g., news articles regarding the injuries after a terrorist attack. It has been suggested that individual user's interests are to some extent consistent with the local news trends [1]. A popularity-based recommender is useful in that situation since it can capture the popularity of news articles quickly. The occurrence of weather problem will influence recommender's performance.

[1]https://www.last.fm/

2.1.5. Discussion

Previous work from NewsREEL Challenge has shown content-based filtering approach does not work well in the news recommendation field [3]. The content-based filtering approach would continuously recommend news articles which are similar to the one user is currently reading. However, the recommended news articles would be of no interest to users since they are probably outdated and users are likely to prefer popular news articles. Collaborative filtering approaches [1, 2] would face a continuously cold-start problem from the item side due to the item churn. In addition, they need to deal with the sparsity problem since most items are visited by users few times and the item set changes continuously, which makes the whole U-I matrix sparser. Furthermore, users' preferences are much harder to model in an unregistered system. Despite these difficulties, previous work from NewsREEL still manages to build a recommender based on item-item collaborative filtering. In the method, only $50,000$ most recent user-item interactions are kept [6] and each time the recommender would recommend similar news articles given the news article the user is currently reading. In addition, a fallback recommender in case of the cold-start problem.

Another innovative recommendation method used in news recommendation is contextual bandit approach [14], in which various contextual bandit algorithms, which are combined with context features from users and items, are implemented to choose the right news articles with the purpose to maximize the total CTR. In their work, they also propose an innovative contextual bandit algorithm that assumes there is a linear relationship between the expected payoff (in other case clicks) and the feature vector containing context information from users and items (see Sec. 2.3.2 for details). The method distributes a small amount traffic to explore the under-explored news articles in the long tail with few interactions with users. We also set up a contextual bandit approach for news recommendation. However, in our settings, the contextual bandit algorithm is trained to build the delegation model rather than to be a recommender itself and chooses appropriate recommender each time a recommendation request come. The objective of the contextual bandit algorithm is more complex in our case.

Since our thesis is carried out in the NewsREEL Challenge framework, here we also discuss a little bit about the nature of the NewsREEL Challenge. In NewsREEL Challenge, recommendations must be given within a time limit 100ms since slower recommendation are not acceptable in real-world settings. Thus, a model-based CF approach is also not applicable in our case due to its high computational cost. In the field, a simple popularity or recency-based recommender works well since users are more likely to be attracted by popular news. Due to the good performance of the popularity-based recommender in the field, our proposed delegation model is built on top of four popularity-based news recommenders and one recency-based recommender. However, in the future work, we would like to try more recommender (see Sec. 6.5.2 for more details).

2.2. Multi-armed Bandit Problem

The multi-armed bandit problem was first introduced in the probability theory. The problem considers a gambler at a row of slot machine deciding how to play those machines in order to maximize the sum of rewards from the sequence of plays [25]. Multi-armed bandits are similar as a room with many slot machines, and each slot machine has a random payoff distribution [26]. Multi-armed bandit algorithms have recently been considered as an alternative to A/B testing for online optimization. A/B testing is the main online testing method in the industry. It is often used in the website optimization [27] field to decide which implementation is better, such as the color of a specific button and the format of ads [28]. In A/B testing, live traffic is divided into groups, the control group and the experiment group, to decide which implementation is the best. A/B testing is a beneficial method when it comes to deciding which design or implementation is the best. However, when the objective is to maximize the reward from the model, it is less promising than the multi-armed bandit algorithms. A/B testing usually consists of a period of a relatively short time of pure exploration and a long time of pure exploitation [27] (See Sec. 2.2.1 Exploitation vs Exploration). During the exploration phase, it would waste much time on inferior options, while during the exploitation phase, it would continue choosing the best options from the exploration phase and never be back to inferior options even if some of them show better performance later. In our case, we decide to build our delegation model based on the multi-armed algorithms and we expect it to improve the recommendation performance (maximize the results from different evaluation metrics, such as CTR).

In our settings, there are several recommenders available. We would like to build a delegation model on top of the recommender using multi-armed bandit algorithm. Each time when there is an incoming recommend request, it would choose a recommender to serve and meanwhile delegate the request to the chosen recommender. The chosen recommender then gives back the recommendation response. Afterwards, on receiving users' feedback (the reward), the multi-armed bandit based delegation model is able to update.

2.2.1. Technical Terms

In this section, we introduce several technical terms from multi-armed bandit algorithms.

- **Arm**

  Options that can be chosen. In our case, arms are the several recommenders with detail description in Sec. 3.1.

- **Reward**

  A measure of success. In the business field, it most of the time related to profit. For example, in the advertisement recommendation, it is usually defined as CTR or revenue triggered by clicks from certain advertisement [29]. In our case, the reward is defined as feedback from users, namely the click behavior. In addition to the click behavior, rewards can also have other definitions depending on what we would like to maximize during the recommendation. In this work, rewards are defined as clicks and has two numerical values, 0 and 1. 0 means no clicks observed during the recommendation, while 1 means there is a click observed from the recommendation (see Sec. 3.2 for more details).

- **Bandit**

  A collection of arms. In our case, a bandit is a collection of different recommenders.

- **Play/Trial**

  Each arm will be pulled multiple times in the experiment. Each pull action is called a play or a trial. In our case, each time the delegation model chooses a recommender to serve is called a play or a trial.

- **Exploitation vs Exploration**

  A multi-armed bandit algorithm exploits if it plays the arm with the highest estimated reward based on its experience of previous plays. Exploration is exactly the other way around, which means the algorithm plays an arm that does not have the highest estimated reward. One main concern of multi-armed bandit algorithm is to make a trade-off between exploitation and exploration to maximize the total rewards.

2.2.2. Multi-armed Bandit Algorithm

In this section, we introduce several commonly used multi-armed bandit algorithms.

- **epsilon-Greedy**

  Epsilon-Greedy is the simplest algorithm to solve the multi-armed bandit problem. The only parameter in the policy is ϵ. In the standard settings, ϵ is fixed. With probability $1-\epsilon$, the policy plays the currently best arm, and with probability ϵ, it plays a random arm. The algorithm also has several variations. In the first variation, ϵ is not fixed and is able to vary with time [30]. With the variation, the algorithm is able to explore more at the beginning and less at the end when it has received enough knowledge from the arms. In this case, ϵ decreases over time. The behavior is called annealing [27], with the strategy called ϵ-decreasing strategy. Another variation of the epsilon-Greedy policy is ϵ-first. To implement the algorithm, the total number of plays H (also known as Horizon) should be known beforehand. In the first ϵH plays (also known as the exploration phase), each time the algorithm would randomly choose an arm to play. While in the later $(1-\epsilon)H$ plays (also known as the exploitation phase), each time it would choose the arm with the best performance based on its knowledge learned from the exploration phase. The algorithm outperforms the standard epsilon-Greedy policy when the horizon is known in advance [30].

- **Softmax**

  One disadvantage of the epsilon-Greedy algorithm is it does not distinguish the difference among the rest inferior arms [27]. The Softmax algorithm considers the difference of the estimated value among different arms. The probability of choosing an arm is proportion to its estimated value. In the standard settings, the probability to play the arm is equal to $e^{\frac{r_\alpha}{\tau}} / \sum_{i=1}^{n} e^{\frac{r_i}{\tau}}$, where r_α is the estimated reward for arm α from previous plays, n is the number of plays, and τ is a parameter which varies from 0.0 to 1.0. Similar to ϵ in epsilon-Greedy, τ is fixed in the standard settings. While in the annealing settings, τ also changes over time, which enables the algorithm to explore less with time going on.

- **Upper Confidence Bound**

  Different from the above algorithms, UCB is not only concerned about the reward from the previous plays but also about how much it knows about the available arms. At the initial phase, UCB explores each arm at least once. After that, each time it plays the arm with the highest value from the formula $r_\alpha + b$, where r_α is the estimated reward for arm α, and b is a special bonus (confidence bound) for the arm α. One widely used UCB variation is UCB1 [31], in which b is set to be $\sqrt{\frac{2ln\sum_{i=1}^{n} t_i}{t_\alpha}}$, t_i is the number of plays arm i has been chosen, and t_α is the number of plays arm α has been chosen. In UCB, arms that are explored less in the previous plays would receive higher bonus value, while arms that are explored more would receive lower bonus value. The algorithm can prevent the under-exploration of potential rewarding arms. Besides, unlike Softmax and epsilon-Greedy, UCB does not use any randomness and its exploration is therefore under the guidance of the confidence bound.

- **Thompson Sampling**

  Another widely used multi-armed bandit algorithm is Thompson Sampling. Thompson Sampling models the probability distribution of each arm's reward. The reward of arm α follows a probability distribution with mean θ_i^* [32]. A Beta-Bernoulli distribution [33] is used when the reward is a binary value. Each time, the algorithms would draw a sample from the distribution of each arm and play the arm with the largest sample value. Thompson Sampling for Bernoulli bandits [33] works in the way below.

Algorithm 1 Thompson Sampling for Bernoulli bandits

```
$S_i$: number of success times, $F_i = 0$: number of failure times
 1: For arm $i = 1, 2, ..., N$, initialize $S_i = 0, F_i = 0$
 2: for trial $t = 1, 2, ...,$ do
 3:     for each arm $i$ do
 4:         $\theta_i(t) = sampleSelection(betaDistribution(S_i + 1, F_i + 1))$// Draw sample $\theta_i(t)$ from the Beta
            Distribution
 5:     end for
 6:     play arm $i$ with $argmax\theta_i(t)$// Plays arm $i$ with the largest sample value $\theta_i(t)$ in trial $t$
 7:     observe the reward of this play
 8:     if reward $r = 1$ then
 9:         $S_{i(t)} = S_{i(t)} + 1$
10:     else
11:         $F_{i(t)} = F_{i(t)} + 1$
12:     end if
13: end for
```

2.3. Contextual Bandit Algorithm

In multi-armed bandit algorithms, the arm-playing decisions are based on the algorithms' knowledge about arms. However, we often have context information available when trying to figure out which arm to play. For example, during advertisement recommendation, we may find some advertisements are more attractive to young users than to old users. Here, age is side information from users and can be considered as a type of context. Context benefits decision-making. Many examples of previous work implement contextual bandit algorithms for decision-making. Tang et al. [28] implement contextual bandit algorithms for automatic advertisement format selection. Li et al. [29] build an advertisement re-ranking model based on contextual bandit algorithms to improve advertisement reach and CTR. In work [14], Li et al. build a news recommendation model based on their new proposed contextual bandit algorithms LinUCB (see Sec. 2.3.2). The basic idea is people from different places with different age and gender may prefer different types of news articles. In the work, users and news articles are represented by feature vectors of categorical features. To the best of our knowledge, it was also the first time that contextual bandit algorithms have been implemented for use in recommendation. As indicated in Sec. 1.4, we also implement contextual bandit algorithms for recommendation purposes. However, the main difference lies in the fact that their goal is to build a recommender based on contextual bandit algorithm making use of side information from users and items to directly recommend news articles to users, our consideration is to build a delegation model based on contextual bandit algorithm to choose appropriate recommender to serve the recommendation requests making use of the context that might influence the recommender's performance. Later, after news recommendation, Linkedin, the employment-oriented social networking website also began to improve their job recommendation through contextual bandit algorithms [34].

Based on how context is combined [22] with the bandit algorithm, we classify the algorithms into two group. A trivial implementation is to build a multi-armed bandit model for each context [28, 29] and we call it context pre-filtering and discuss it in Sec. 2.3.1. Another idea assumes there is a relationship between the expected reward and the context [14]. The method is called context modeling and is discussed in Sec. 2.3.2. Here, context refers to feature vectors of categorical features. In our case, the categorical features can be side information from users, news articles as well as the external environment, such as weekend or weekday information during the recommendation.

2.3.1. Context Pre-filtering

In the context pre-filtering method, a multi-armed bandit model is built for each context. Here, context can be feature vectors containing side information from both users and items. It is a trivial implementation and shows good performance in previous work [14]. In previous work [14], context is represented as feature vectors that contain users' attributes and items' attributes, such as users' gender and age information. Some dimension reduction methods are then used to reduce the dimension of the feature vectors with the feature vectors at last grouped into several clusters. In their implementations, a multi-armed bandit model is built in each cluster and is responsible for choosing the right news article to recommend each time. The multi-armed bandit models in their implementations are basically served as news recommenders. However, the multi-armed bandit models serve a different purpose in our case. They are implemented to choose the appropriate recommender for each recommendation requests. In our case, only one type of context (*Domain*) is used in the context pre-filtering method. The context has three different categorical values and is encoded as feature vectors using one-hot encoding (see Sec. 3.3.3 for details about the context features used and Sec. 4.4 for encoding details).

2.3.2. Context Modeling

Another method we consider is context modeling. It assumes there is a relationship between the expected reward and the context [14]. In our case, contextual information is first represented as feature vectors (using one-hot encoding, see Sec. 4.4 for details) and then combined with the bandit algorithm to obtain the estimated reward. In addition, two types of context, *Domain* and *User-Domain Impressions* are used in our implementations (see Sec. 3.3.3 for details about the context features used).

- **LinUCB**

  LinUCB is a contextual bandit algorithm first proposed in [14]. The algorithm assumes that there is a linear relationship between the expected reward and the context. In this algorithm, the context is used to update the rewards and improve its arm selection strategy. In each trial (play) t, the expected payoff of arm α is equal to $E[r_{t,\alpha}|x_{t,\alpha}] = x_{t,\alpha}\theta^*_\alpha$, where $r_{t,\alpha}$ specifies the reward of arm α at trial t, $x_{t,\alpha}$ specifies the context feature vector for arm α at trial t and θ^*_α is a coefficient that can be learned from the previous trials. In each trial, the algorithm plays the arm with the highest final score. The final score of each arm is calculated as:

$$\alpha_t = \text{argmax}_{\alpha \in \mathtt{A}} \left(x_{t,\alpha}\hat{\theta}_\alpha + \beta\sqrt{x_{t,\alpha}^T A_\alpha^{-1} x_{t,\alpha}} \right) \tag{2.1}$$

where $\mathtt{A}$ is the set of arms, $\hat{\theta}_\alpha$ is an estimate of coefficients, β is a hyper-parameter and A is the covariance matrix of coefficients.

2.4. Evaluation Metric

Evaluation is an important part of recommender systems. In some recommendation fields, e.g., music recommendation and movie recommendation, where ratings from users are available, prediction accuracy metrics such as Root Mean Square Error (RMSE) are always used. This evaluation metric is not applicable in news recommendation since there are no explicit ratings in news recommendation. In this section, we mainly focus on the evaluation metrics that are applicable to the news recommendation field. The evaluation metric provided by the Challenge consider both companies' needs and users' needs. Companies like Plista [2] and the underlying news portals (see Table 5.1) usually care about the Click Through Rate from which they gain revenue. Users care about (1) whether the recommendations satisfy their information needs, (2) whether the recommendations are provided within a tolerable time limitation.

2.4.1. NewsREEL Evaluation Tasks

The NewsREEL Challenge allows participants to evaluate their recommender performance in tasks [10]. In the first task, NewsREEL Live, real-time user requests from different publishers are redirected by Plista, to the participants. The participants need to provide well-formed recommendations to the incoming requests within a time limit. The three requirements are: (1) a certain number of recommendations must be generated, (2) the generated recommendations must be provided within a time limit (100ms in our case), (3) the response containing the generated recommendations must be delivered in a valid format. Together with the three requirements, the official quantitative evaluation metric is defined as:

$$Online_{CTR} = N_{click}/N_{request} \tag{2.2}$$

where N_{click} is the number of clicks from the generated recommendations and $N_{request}$ is the total number of recommendation requests sent to the system.

The second task, NewsREEL Replay, is a near-online evaluation. A one-month dataset is first collected and then replayed chronologically by a simulator to simulate the stream data. The official quantitative evaluation metric is called prediction accuracy (offline CTR). It is similar to the evaluation metric defined in the Live task:

$$Offline_{CTR} = N_{successful}/N_{total} \tag{2.3}$$

where $N_{successful}$ is the number of successful recommended items and N_{total} is the total number of items recommended. One problem in the Replay Task is users are not able to see the recommended items and thereby cannot click on the items. The problem is solved in the following ways: given a recommended item generated by the recommender, if the user interacts with the item in the next 10 minutes, it will be considered as a successful recommended item. More details regarding the Live and Replay evaluation tasks will be explained in Chapter 5.

In summary, CLEF-NewsREEL Challenge provides participants with an opportunity to evaluate their recommender in more than one dimension. CTR is what companies like Plista and news portals care most about. In addition, CTR also reflects users' satisfaction with the recommendation. If the target user is interested in the recommended news article, a 'click' event will be triggered. Furthermore, the evaluation methods provided by the challenge also covers two other aspects: (1) response time, and (2) response validity. CTR, response time, and response validity cover three evaluation dimensions in both Live Task and Replay Task.

[2]https://www.plista.com/

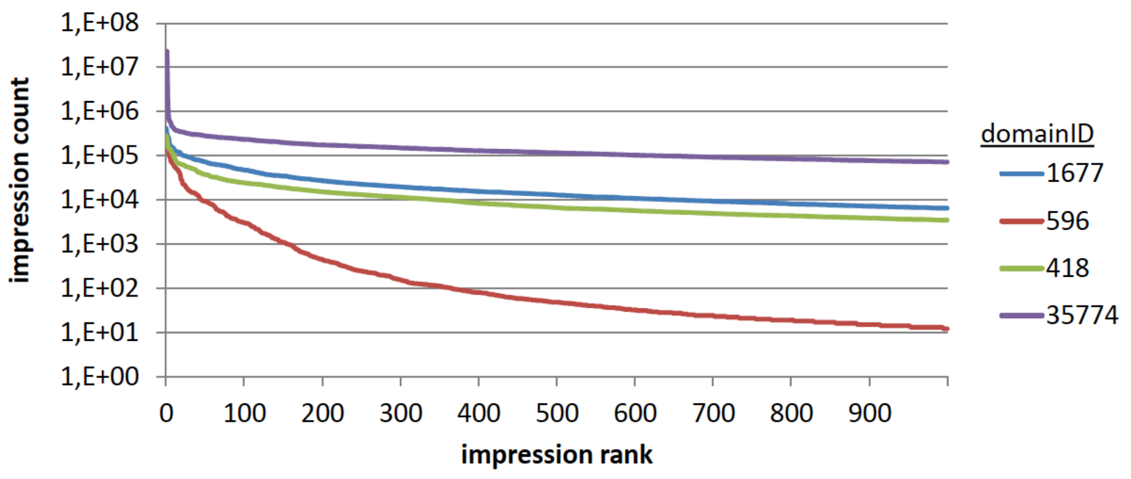


Figure 2.1: Illustration of updates and the end of each batch, source from [8]
Notes: Impression count measures how many times a news article has been visited. The higher the impression count is, the lower the impression rank is. Different news portals are represented by different *domainID*. For more details about *domainID* and its corresponding publishers, see Table 5.1

2.4.2. Catalog Coverage

CTR is the most widely used metric in online evaluation [35]. In the online news recommendation scenario, companies' profit is directly related to users' overall CTR, and from the perspective of users, users' click behaviors can reflect their interests. However, we argue that CTR considers more the company needs and should not be the only metric to measure the recommendation performance from the user side. For example, simple popularity-based recommender shows high overall CTR performance in CLEF NewsREEL Challenge news recommendation settings [3, 8]. Popularity-based recommender promotes news articles with high impression counts. One example of previous work from the NewsREEL Challenge analyzes the distribution of the impression counts over the whole set of news articles [8]. The figure below shows the number of impressions with respect to news articles. News articles with the highest impression count have the top position. We can see that few news articles have very high impression counts, suggesting these articles are of high interest for most users. Popularity-based recommender contributes to overall CTR by always recommending the popular articles, which at least capture a part of the interests of most users. However, the fact that news articles from the long tail are also clicked suggests that users are not always interested in popular news articles. Their needs for the interested but unpopular news articles are somehow ignored in this case. Popular-based recommender seems to be a promising recommender for people who enjoy popular news articles, but not a good recommender for those who are looking for something of their interests but not popular. Also, from the editor's perspective, they would be willing to see more distinct news articles to be recommended.

Therefore, to recommend more distinct news articles instead of popular ones, we also consider another metric called coverage [18, 36] to evaluate our system's performance on recommending items in the long tail. Coverage measures the fraction of items which the recommender would recommend to the total item set (in our case, items that users ever interacted with). To the best of our knowledge, no other previous work in the online news recommendation field has tried the metric before. It is defined as the ratio of distinct items recommended across the whole dataset, and in our case, can be expressed as:

$$CC = N_{distinct}/|I| \tag{2.4}$$

where $N_{distinct}$ is the number of the distinct items in the recommendation and $|I|$ is the total number of items in the dataset. In our work, catalog coverage is only used in the near-online scenario and is measured over the two-day data collected from day 2016-02-02 to 2016-02-03 (see Sec. 5.1.2 and Sec. 5.1.4 for details about the dataset used in the near-online scenario).

3 Delegation Model

In this chapter, we describe our delegation model with respect to three aspects: (1) news recommender algorithms from which the delegation model can choose from, (2) delegation framework, (3) context that may influence recommender's performance and how context is combined with the delegation model

3.1. News Recommender Algorithms

The delegation model is built on top of five news recommenders. The original code of these recommenders is developed by recommenders.net and can be found in the two Github repository [37, 38].

- *Recent*: Recommend the most recently requested news article. A recommendation request is generated each time a user lands on a news webpage. Here, the requested news article is the news article from the news webpage. For the realization, a ring buffer is implemented to store the most recently created or updated articles. The size of the ring buffer is set to 100. When there is a new item, it will be inserted into the ring buffer. If the ring buffer reaches its size, it will drop the least recent news articles.
- *Path*: Given the news articles that the user is currently reading, recommend the most popular[1] news articles requested next by users, referring to the *Most Popular Sequence* in the work [7]. In the default settings, 100 most recently requested articles following the current news article are kept. The popularity scores are then computed from the 100 articles.
- *Click*: Similar to the algorithm *Path*, but the popularity algorithm considers only the clicks. It should be clear that a 'click' event is different from an impression: a 'click' event means the user clicks on the recommended news articles, while an 'impression' event means the user lands on certain news article page but it is not necessarily a recommended one [3].
- *Imp*: Similar to the algorithm *Path*, but the popularity algorithm considers only the impressions from users. An impression will be created if a user lands on a news article page.
- *PRCate*: Popularity and category based Recommender. Recommend the news article with most impressions and clicks within a certain category. For a general news portal, categories are 'sports', 'entertainment', 'politics', etc. While for a sports news portal, categories can be specified as 'football', 'handball', 'basketball', etc. The popularity scores only consider 100 most recently requested news articles within the category.

3.2. Delegation Model Work Flow

The section explains how the delegation model is designed and how it works. Although in our work, the delegation model is only trained and updated in the near-online evaluation due to the low data volume received in the online evaluation (See Sec. 5.3.2 for details), it should work in an online scenario given

[1]The popularity is computed from the weighted combination of impressions and clicks.

enough data. The delegation framework is a general framework and should work in both near-online and online evaluation.

Implementing a delegation model based on the multi-armed bandit algorithm is not trivial in real-world settings. Firstly, we need a measure of recommendation success, namely a definition of rewards. As shown in Sec. 2.2.1, rewards in our work are defined as clicks from users. A reward records whether the current user clicks on the recommended item and has two numerical values: 1 and 0. In addition to clicks, rewards can also have other definitions. The definition of rewards depends on what we would like to maximize during the recommendation. It can be clicks, revenues from the clicks and catalog coverage. For more discussion about the reward definition, please refer to Sec. 6.5.3.

However, unlike what is implemented in [14, 29], where rewards can be obtained immediately after each action, rewards in real-world settings are always delayed. Specifically, in our case, there would be hundreds of incoming recommendation requests between a recommendation and its corresponding reward, the user's click. It is unrealistic to suspend the delegation model before receiving the reward or to update the whole delegation model each time a reward is observed. To solve the problems, we implement a batch update similar to what is described in the work [28]. With the batch update, the delegation model can update in batches. It updates only after accumulating enough rewards.

3.2.1. Batch update and assignment

During the recommendation process, the delegation model only updates at the end of each batch with the rewards collected in the batch, as illustrated in Figure 3.1. Then, it will pre-assign appropriate recommenders for each context to serve incoming requests in the next batch. Here, context can be *Domain* (in context pre-filtering) or a combination of *Domain* and *User-Domain Impressions* (in context modeling). The main purpose of the pre-assignment is to reduce the delegation time.

3.2.2. Initialization

Before the first batch, there is no historical data in the delegation model. It knows nothing about each recommender. To provide initial training data, recommenders are randomly selected to serve the incoming recommendation requests in the first batch. At the end of the first batch, the delegation model is able to update for the first time with the rewards collected in the batch.

3.2.3. Rewards

In order to improve the CTR, rewards are defined as 'clicks' from the recommendations. However, in both tasks, we are not able to track the click behaviors following the recommendations. In the near-online settings, for a given recommendation, click behaviors are defined as whether the user interacts with the recommended item in the next 10 minutes. Users' interaction with the recommended item in the next 10 minutes is a simulation of real click behaviors which is also utilized in NewsREEL Replay.

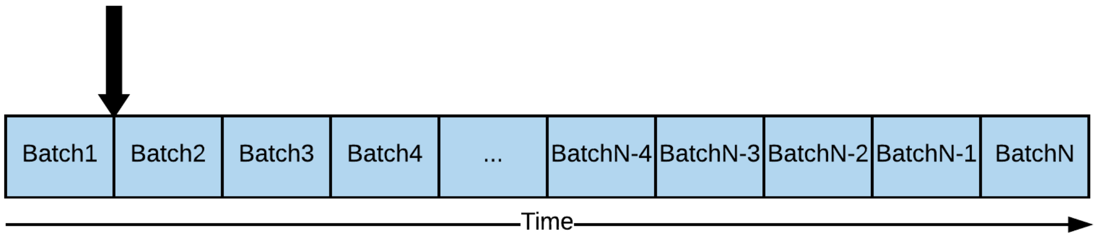


Figure 3.1: Illustration of updates and the end of each batch

3.3. Exploiting Context

As indicated in Sec. 2.3.1 and Sec. 2.3.2, there are two ways to combine context with multi-armed bandit algorithm. Here, context is a set of feature vectors which are encoded in the incoming recommendation requests and containing side information about users, news articles and the external environment, e.g. the time of the day. To make use of the context, we further transform the feature vectors by one-hot encoding (see Sec. 4.4 for details). The first way is to build a multi-armed bandit model for each context and is called context pre-filtering (see Sec. 2.3.1). While the other way is to combine context directly with the bandit algorithm to obtain the estimated reward and is called context modeling (see Sec. 2.3.2). Based on these two ideas, two different types of delegation model are built. However, before talking about how contextual bandit delegation model works, we would like to first introduce how context-free delegation model work. In the first step, we initialize hyper-parameter in the algorithm that is used to balance the exploration and exploitation trade-off. Then, for each incoming recommendation request, the delegation model chooses a recommender to serve the requests. Next, the chosen recommender gives back the recommendation response. Based on whether the user clicks on the recommendation, the reward is assigned as 0 or 1. With the reward, the whole delegation model is then able to update. It should be noticed that in real-world settings, the reward is always delayed and the delegation model always updates in batches. However, to keep it simple for illustration reason, we assume there are no delayed feedbacks and the delegation model updates each time after receiving the reward. The algorithms below show how different types of delegation model work in each batch.

Algorithm 2 Context-free delegation model: multi-armed bandit algorithm based

recs: different recommenders, *rec_response:* recommendation response, *rec_request:* recommendation request, *interaction:* user-item interaction, *historyInteraction:* user-item interaction history

```
1: Initialize the hyper-parameter, e.g. $\epsilon$ in epsilon-Greedy
2: for $each\ rec\_request$ do
3:   $rec \Leftarrow recommenderSelect(recs, historyInteraction)$
4:   $rec\_response \Leftarrow playRec(rec)$
5:   if $recs\ is\ clicked$ then
6:     $reward \Leftarrow 1$
7:   else
8:     $reward \Leftarrow 0$
9:   end if
10:  $updatePolicy(rec, reward)$
11:  $updateHistory(interaction, historyInteraction)$
12: end for
```

3.3.1. Delegation Model based on Context Pre-filtering

Our first approach builds a delegation model for each context. For example, build a delegation model for each publisher from which the recommendation requests come. The model generally works in a similar way as the context-free delegation model. However, each time when there is an incoming request, the related context need to be extracted from this request first. The delegation model built for the context is then responsible for choosing the appropriate recommender to serve. Also, only delegation model built for the context can update later. The algorithm of the delegation model can be found in Algorithm 3.

3.3.2. Delegation Model based on Context Modeling

Our second approach exploits context for training the delegation model directly and is different from context pre-filtering. Here, context is integrated directly into the delegation model. We use the Linear UCB algorithm [14] to exploit context in the model. We use a special case of general contextual bandit algorithm known as *K-armed bandit*, where the set of arms remains the same in all trials. In our framework, the five recommenders introduced earlier in this section serve as the arms. The score of arms is calculated according to Eq.(2.1) where the feature vectors $x_{t,\alpha}$ are built based on the given context. The algorithm of the delegation model can be found in Algorithm 4.

3.3.3. Context Features

In this section, we discuss the context that we used in our delegation model. Here, we use the context that can be extracted from the incoming message directly. For our future work, we might try more

Algorithm 3 Context pre-filtering based delegation model

recs: different recommenders, *rec_response:* recommendation response, *rec_request:* recommendation request, *interaction:* user-item interaction, *historyInteraction:* user-item interaction history, *rec_contextX:* context X extracted from the incoming recommendation requests

```
1:  Initialize the hyper-parameter, e.g. $\epsilon$ in epsilon-Greedy
2:  for $each\ rec\_request$ do
3:    $rec\_contextX \Leftarrow extract(rec\_request)$
4:    $rec \Leftarrow recommenderSelect(recs, rec\_contextX, historyInteraction\_X)$
5:    $rec\_response \Leftarrow playRec(rec)$
6:    if $recs\ is\ clicked$ then
7:      $reward \Leftarrow 1$
8:    else
9:      $reward \Leftarrow 0$
10:   end if
11:   $updatePolicy(rec, reward, rec\_contextX)$
12:   $updateHistory(interaction, historyInteraction\_X)$
13: end for
```

Algorithm 4 Context modeling based delegation model

recs: different recommenders, *rec_response:* recommendation response, *rec_request:* recommendation request, *interaction:* user-item interaction, *historyInteraction:* user-item interaction history, *rec_contextX:* context X extracted from the incoming recommendation requests

```
1:  Initialize the hyper-parameter, $\beta$ in LinUCB
2:  for $each\ rec\_request$ do
3:    $rec\_contextX \Leftarrow extract(rec\_request)$
4:    $rec \Leftarrow recommenderSelect(recs, rec\_contextX, historyInteraction)$
5:    $rec\_response \Leftarrow playRec(rec)$
6:    if $recs\ is\ clicked$ then
7:      $reward \Leftarrow 1$
8:    else
9:      $reward \Leftarrow 0$
10:   end if
11:   $updatePolicy(rec, reward, rec\_contextX)$
12:   $updateHistory(interaction, historyInteraction, rec\_contextX)$
13: end for
```

context and even context combined with temporal information (see Sec. 6.5.1 for more discussion).

What kind of context can we use? Unlike some of the previous work [14, 28] where user-profiles are available, here, we don not have any information about user profiles. Users can only be tracked by their cookies, and some users even do not allow any behavior tracking. All contextual information that we can use is encoded in the incoming messages. The description of the context available in the incoming message can be found in the ORP Protocol [39]. However, the documentation is not detailed when describing the context features due to confidential reasons, which also poses a challenge to us.

Not all context features can contribute to the performance of recommender algorithms, such as some browser related context and the weather of the day. In our work, we consider two context features in total. One is called *Domain*, which describes from which publisher the certain recommendation request comes, and the other is called *User-Domain Impressions*, which refers to the number of impressions the user has with the domain. The context *Domain* is considered as an important context feature in the work [7]. In the implementation, context is encoded as feature vectors using one-hot encoding (see Sec. 4.4 for details).

- **Domain**

Figure 3.2 shows the recommender algorithms' performance with respect to different context features using the CTR metric. In domain '418' (*ksta*, a general news portal), the *Click* recommender outperforms other recommenders. It also shows good performance in domain '1677' (*tagesspiegel.de*, also a general news portal). The *Recent* recommender performs best in domain '35774' (*sport1.de*, a sports news portal). People from sports portal prefer the latest sports news requested by others. Popularity and category-based recommender (*PRCate*) generally performs better in the two general news portals than in the sports portal, where the recommender shows a very low CTR performance. This might be due to the fact that the sports portal is already very specific, and further specific category would not help. The findings are consistent with the previous work in NewsREEL Challenge.

- **User-Domain Impressions**

  Figure 3.3 and 3.4 shows the algorithms performance with respect to the number of user-domain impressions in domain '418' (*ksta*) and '1677' (*tagesspiegel.de*) respectively. Normally, with the increase of the number of user-domain impressions, CTR should also be increased as users with more interactions with the publishers would be more willing to click on the recommendations. However, there are some exceptions. With the increase of *User-Domain Impressions*, CTR observed from the *Click* algorithm experience a decrease in Figure 3.3. The decrease can be explained by the fact people with enough interactions with the publisher have already seen the articles recommended before.

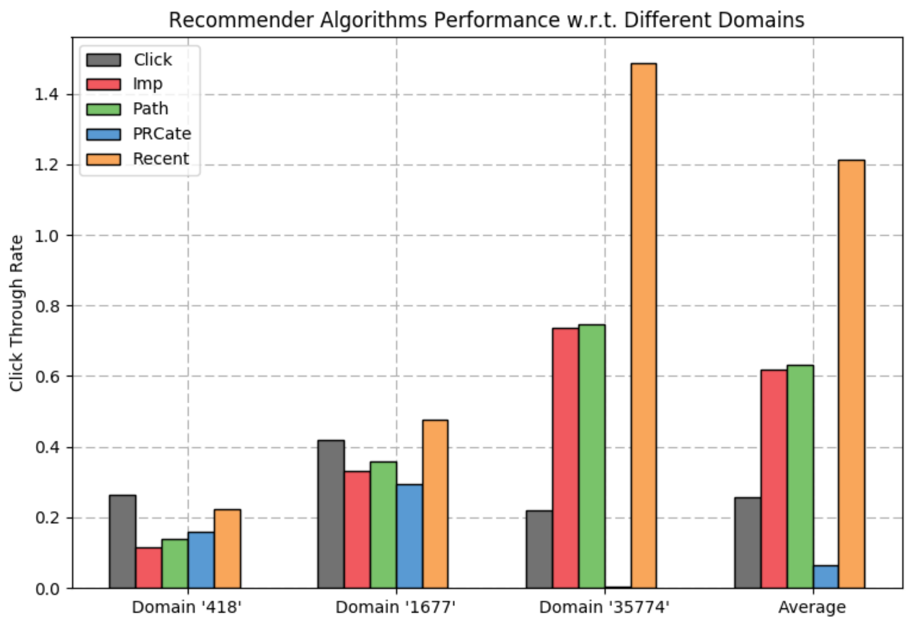


Figure 3.2: Recommender Algorithms Performance with respect to Different Domains

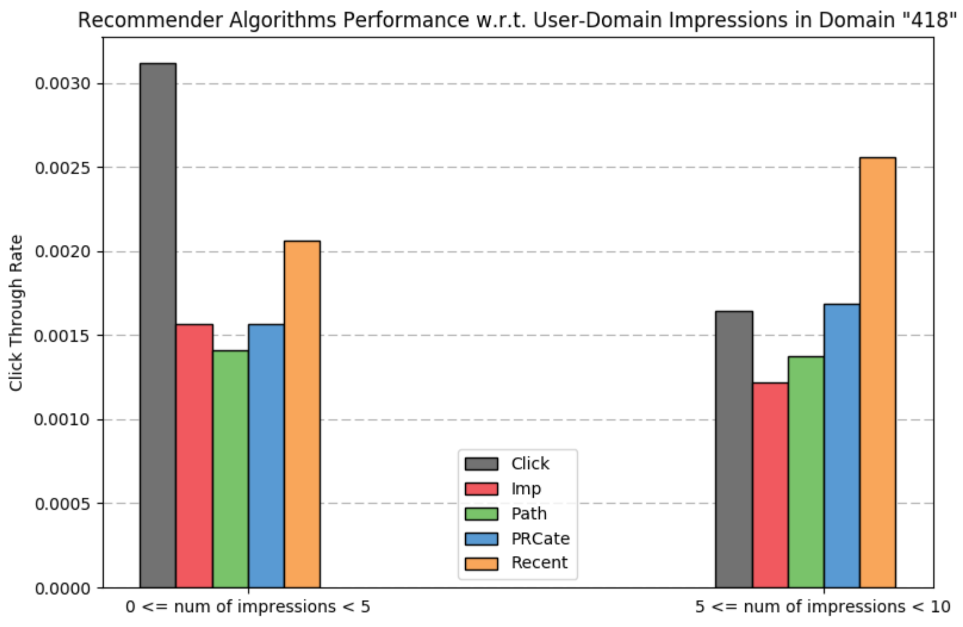


Figure 3.3: Recommender Algorithms Performance w.r.t. User-Domain Impressions in Domain "418"

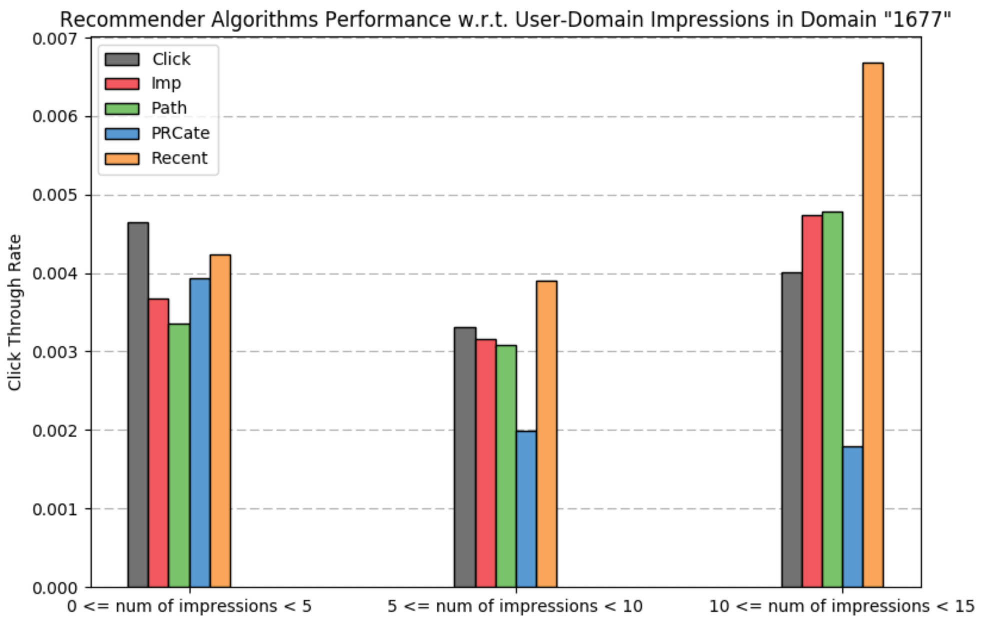


Figure 3.4: Recommender Algorithms Performance w.r.t. User-Domain Impressions in Domain "1677"

4 Implementation

This chapter clarifies the implementation details of our recommender.

4.1. Open Recommendation Platform

The Open Recommendation Platform (ORP) is a distributed platform which is responsible for handling the recommendations in the online scenario [39]. Recommendation consumers and providers are able to interact through the platform over a standardized protocol, ORP protocol. To communicate with the ORP Platform, we utilize the Java SDK[1] provided by the organizer of NewsREEL Challenge.

4.2. Server Architecture

This section introduces our recommendation server, delegation model and implementation details about how the recommendation server and delegation model is built technically.

4.2.1. Recommendation Server

Our recommendation server is built on SurfSara Virtual Machine[2]. Figure 4.1 shows the structure of our recommendation server. It consists of recommender client, various recommenders, and delegation model built on top of the recommenders. The recommender client is set for communication purposes. It is responsible for processing the incoming messages from users and then delegating the processed messages to the delegation model. The delegation model, after receiving the recommendation request, will then delegate it to the appropriate recommender. The chosen recommender then gives back the recommendation response to the recommender client and further back to the user.

4.2.2. Delegation Model

In the offline scenario, rewards are first obtained at the end of each batch making use of the offline evaluator. Then, the contextual bandit based delegation model is able to update with the rewards. Next, a sequence of recommenders is assigned in block beforehand to serve the incoming requests from the next batch in order to save the delegation time in the next batch.

4.2.3. Implementation Details

The recommender client is set up by recommenders.net [3] with the code published in the corresponding Github repository [37]. With the recommender framework [38] provided by the NewsREEL Challenge, we build up several simple news recommenders. Furthermore, we implement several delegation models which are responsible for delegating the incoming recommendation request to the appropriate news recommender based on the context extracted from the request and encoded as feature vectors using

[1]https://github.com/plista/orp-sdk-java
[2]https://doc.hpccloud.surfsara.nl/
[3]http://recommenders.net/

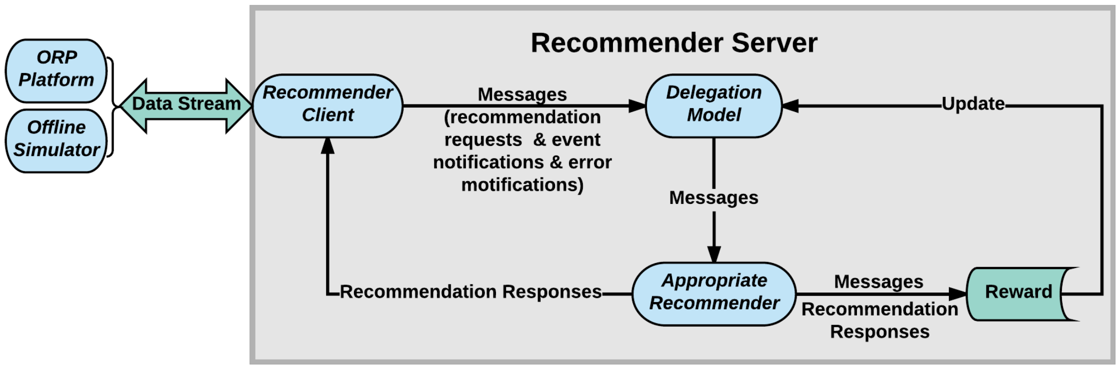


Figure 4.1: Recommender Sever

one-hot encoding (see Sec. 4.4). The delegation model is built based on the contextual bandit algorithms discussed in Sec. 2.2.2 and Sec. 2.3 with two Python libraries [40, 41]. In addition, to calculate the rewards within each batch, we implement a reward-finding method to efficiently find the rewards. Moreover, we implement an evaluation method to calculate the catalog coverage in the near-online scenario.

4.3. Message Types

Here, we clarify four types of messages [39] redirected to our recommendation server during the recommendation.

- **Recommendation requests**

  Recommendation requests are the main type of messages sent by Plista to request the recommendations for a user. A recommendation request is generated each time a user lands on a news webpage. The context information of the user and the news article the user is currently reading is also encoded in the recommendation request. In addition, it also defines how many recommendations should be returned in the corresponding recommendation responses.

- **Event notification**

  Event notifications are sent by Plista with the purpose to announce an event occurring on the ORP platform. Like recommendation requests, context information from users and news articles are also encoded in the messages. There are three types of different event notifications.

  - *impression*: User lands on the webpage with recommendations shown
  - *impression empty*: User lands on the webpage with no recommendations shown
  - *click*: User clicks on a recommendation

- **Item updates**

  Item update messages are sent by Plista to notify the recommender provider about the newly created or updated news articles in the field. The message also encodes a list of context features of the article, including *item id*, *publisher id*, *created time*, *updated time*, *flag* which is used to indicate whether or not the articles should be delivered, *title*, *article text*, *URL*, *image URL*, and *current version*.

- **Error messages**

  Error notifications are sent to notify errors occurring on the server, such as content error (the given context is invalid) or connection errors.

4.4. Combine the Context

In this section, we explain how context is combined with our delegation model. As discussed in Sec. 3.3, we propose two different delegation models based on how context is used in the bandit algorithms. In context pre-filtering, a delegation model is built for each context. While in context modeling, as defined in Eq. 2.1, context features need to be encoded feature vectors. Here, we consider two different types of context features: (1) *Domain* (2) and *User-Domain Impressions*. As *Domain* is already a categorical feature by itself, it is encoded by one-hot encoding. For the context *User-Domain Impressions*, we implement some transformations. Specifically, we classifies the number of *User-Domain Impressions* into three groups: $0 \leq$ number of *User-Domain Impressions* < 5, $5 \leq$ number of *User-Domain Impressions* < 10 and $10 \leq$ number of *User-Domain Impressions* < 15. Then the transformed categorical feature is encoded using the one-hot encoding. For example, the categorical feature {*Domain*: 'ksta', *User-Domain Impressions*: '2'} is encoded as the feature vector $[0, 0, 1, 1, 0, 0]$ in our case. Additionally, the context *Domain* has three categorical values, corresponding to the three different domains: *sport1.de*, *tagesspiegel.de* and *ksta*. Although there are eight domains (publishers) from Table 5.1.2 available in the offline dataset, we only consider the three domains since most of the traffic is from these domains.

5 Evaluation

5.1. NewsREEL Replay Near-Online Evaluation

The evaluation part of the project is mainly based on the two evaluation tasks provided by CLEF NewsREEL Challenge 2017: NewsREEL Replay Near-Online Evaluation and NewsREEL Live Online Evaluation.

5.1.1. Near-Online Evaluation Structure

The near-online evaluation method provides us with a way to evaluate our recommender in a near-online scenario by simulating the recommendation task happening in the real world. The near-online recommendation consists of four components: Dataset, Sender, Recommender and Evaluator. Figure 5.1 below shows the whole recommendation and evaluation process. Firstly, the sender is responsible for replaying the messages (see Sec. 4.3) collected in the dataset (see Sec. 5.1.2) to simulate the stream recommendation task in the Live Task. It is able to replay the data through HTTP to the IP address where the recommender deployed. Also, it can record and log the recommendation response back from the recommender for further evaluation use. The recommender, on receiving the recommendation requests, must provide a valid recommendation response and send the response back to the sender. After this recommendation simulating process, the evaluator is responsible to evaluate recommendation response logged by the sender against the original user behaviors from the dataset.

As explained in Sec. 2.4.1, no 'Click' events can be generated during the evaluation process. Instead, the offline *CTR* (see Eq. 2.3) is defined as the main evaluation metric in the near-online evaluation. It considers a user's interaction with the recommended items within a 10 minutes' time window after receiving the recommendation as a 'Click' event. The evaluation results of near-online CTR are shown in Sec. 5.1.5. Another evaluation metric inside the near-online evaluation is *Response Time*. It measures the time interval between the recommendation request and the recommendation response. The metric is also essential in our case since a delayed recommendation request will not be accepted by users in a real-world news recommendation scenario. The metric is also consistent with the requirement in the online evaluation, in which recommendation responses must be given within $100ms$. In addition to the evaluation metric provided by NewsREEL Challenge, we also consider another metric, catalog coverage as defined in Sec. 2.4.2 to evaluate the fraction of items recommended by the recommender to the total item set. The evaluation results of response time and catalog coverage from a near-online scenario are shown in Sec. 5.1.6 and Sec. 2.4.2 respectively.

5.1.2. Dataset

Dataset [43] replayed in the near-online evaluation is provided by Plista, the company behind the challenge. A month-long dataset collected during February 2016 serves as the near-online evaluation dataset. The dataset captures interactions from 8 publishers, including two million notifications, 58 thousand item updates, and 168 million recommendation requests. Table 5.1 below shows the statistical summary of the events from the eight publishers across February. As indicated in the table, the dataset is not evenly distributed: about 70% - 75% traffic is from the domain *sport1.de*, followed by *tagesspiegel.de* and *ksta*. In this evaluation phase, we only consider the results from the above three domains since most traffic comes from these three domains. However, only a small subset which

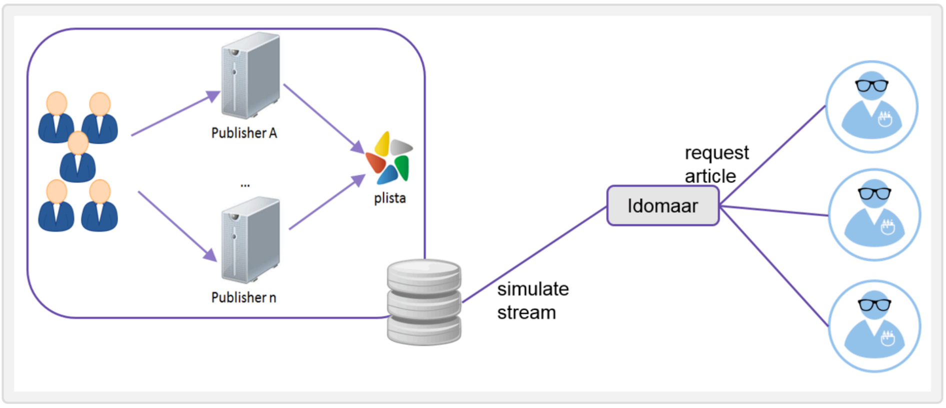


Figure 5.1: Near Online Evaluation Structure, source from [42]

contains traffic from 2016-02-01 to 2016-02-03 is used in the near-online evaluation due to the time constraint (See Sec. 5.3.1). The subset contains $19,641,129$ recommendation requests, $251,796$ clicks and $5,727$ item updates.

Table 5.1: **Dataset Summary**

Domain	Number of Requests	Number of Clicks	Topic
sport1.de (35774)	$126,363,163$	$1,618,300$	sports
tagesspiegel.de (1677)	$22,836,121$	$432,141$	general
ksta (418)	$12,679,105$	$213,114$	general
motor-talk.de (13554)	$9,888,188$	4	automotive
gulli.com (694)	$3,091,277$	$14,749$	IT
3336	$8,411$	46	unknown
2522	60	0	unknown
596	1	1	unknown

5.1.3. Delegation Strategy

For each incoming request from the stream simulator Figure 5.1 here, the delegation model is responsible to delegate the requests to appropriate recommender based on its delegation strategy. In our experiment, the delegation strategies can be classified into three types:

(a.) No delegation strategy used

There is only one single recommender for the delegation model to choose from. The delegation model therefore always chooses the only one news recommender as the best-suited recommender. Evaluation results from the single recommender consists a strong baseline in the near-online evaluation. Single recommender algorithms used in the experiments are the five news recommenders described in Sec. 3.1: *Path*, *Recent*, *Click*, *Imp* and *PRCate* with evaluation results shown in Table 5.2.

(b.) Delegation strategy based on randomness

A delegation model is implemented and able to choose from the five news recommenders mentioned above. However, it does not learn anything from the past. Instead, the delegation model randomly chooses one news recommender from the recommender pool each time there is a new incoming request. The randomness-based delegation model also constructs a baseline in the evaluation and is referred as *Random* in Table 5.2, Table 5.3 and Table 5.4.

(c.) Delegation strategy based on contextual bandit: context pre-filtering

The delegation model is trained using the context pre-filtering method as discussed in Sec. 3.3.1. In the experiment, the only context used in this delegation model is 'Domain' (see Sec. 3.3.3). The corresponding results of the model can be found in Table 5.3. In this delegation model, we experiment several multi-armed bandit algorithms with different hyper-parameters (see Sec. 2.2.2) and different batch sizes (see Sec. 3.2.1):

- epsilon-Greedy with $\epsilon = 0.5$ and $300K$ line messages each batch
- epsilon-Greedy with $\epsilon = 0.3$ and $300K$ line messages each batch
- epsilon-Greedy with $\epsilon = 0.1$ and $300K$ line messages each batch
- UCB1 with $300K$ line messages each batch
- Thompson Sampling with $300K$ line messages each batch
- epsilon-Greedy with $\epsilon = 0.5$ and $100K$ line messages each batch
- epsilon-Greedy with $\epsilon = 0.3$ and $100K$ line messages each batch
- epsilon-Greedy with $\epsilon = 0.1$ and $100K$ line messages each batch
- UCB1 with $100K$ line messages each batch

As explained in Sec. 2.2.2, the hyper-parameter ϵ varies between 0 and 1 and is used to balance the exploration and exploitation. In real-world settings, ϵ is often set below 0.5 to avoid too much exploration.

(d.) Delegation model based on contextual bandit: context modeling

The delegation model is trained based the context modeling method Linear UCB (see Sec. 3.3.2). In the experiment, context used in this delegation model is 'Domain' and 'User-domain Impression' (see Sec. 3.3.3). The results of the model are shown in Table 5.4. The Linear UCB based delegation model is also tested with different hyper-parameters (β is the hyper-parameter defined in Eq. 2.1, which varies between 0 and 1 and is used to balance the exploration and exploitation) and combined context.

- Linear UCB with $\beta = 1$ and $300K$ line messages each batch combined with the context 'Domain'
- Linear UCB with $\beta = 0.5$ and $300K$ line messages each batch combined with the context 'Domain'
- Linear UCB with $\beta = 0.2$ and $300K$ line messages each batch combined with the context 'Domain'
- Linear UCB with $\beta = 1$ and $300K$ line messages each batch combined with the context 'Domain' and 'User-domain Impression'
- Linear UCB with $\beta = 0.5$ and $300K$ line messages each batch combined with the context 'Domain' and 'User-domain Impression'
- Linear UCB with $\beta = 0.2$ and $300K$ line messages each batch combined with the context 'Domain' and 'User-domain Impression'
- Linear UCB with $\beta = 1$ and $100K$ line messages each batch combined with the context 'Domain'
- Linear UCB with $\beta = 0.5$ and $100K$ line messages each batch combined with the context 'Domain'
- Linear UCB with $\beta = 0.2$ and $100K$ line messages each batch combined with the context 'Domain'
- Linear UCB with $\beta = 1$ and $100K$ line messages each batch combined with the context 'Domain' and 'User-domain Impression'
- Linear UCB with $\beta = 0.5$ and $100K$ line messages each batch combined with the context 'Domain' and 'User-domain Impression'
- Linear UCB with $\beta = 0.2$ and $100K$ line messages each batch combined with the context 'Domain' and 'User-domain Impression'

5.1.4. Evaluation Phase

The evaluation phase follows the steps as discussed in Sec. 3.2. Normally, a replay of a daily dump takes about 2 hours. Due to the time constraints, the three-day data from day 2016-02-01 to 2016-02-03 is replayed in the evaluation.

a. The daily dump on day 2016-02-01 is first replayed to the recommender to prevent the cold-start problem

b. The rest of the dataset (collected from day 2016-02-02 to 2016-02-03) is split into smaller mutually exclusive datasets with the same size. The datasets are replayed in batches to the recommender. At the end of each batch, the delegation model is able to update based on its actions during the batch and corresponding rewards received within the batch. In addition, the delegation model will pre-assign a list of recommenders for the incoming requests in the next batch.

For all the delegation models, we measure their performance based on their average CTR across the two days (from 2016-02-02 to 2016-02-03) in the three domains: *sport1.de*, *tagesspiegel.de* and *ksta*. The average CTR is computed from Eq. 2.3. To make sure the delegation model is able to provide valid recommendation within a time limit, we also evaluate the recommender regarding to its response time (see Sec. 5.1.6).

5.1.5. Near-online Click Through Rate

Our first considered evaluation metric is CTR. Table 5.2 shows the *CTR* performance from the randomness-based delegation model and the five news recommenders (see Sec. 3.1) with delegation strategy (a.) and delegation strategy (b.) defined in Sec. 5.1.3. The *CTR* performance of the context pre-filtering based delegation model (referring to delegation strategy (c.) in Sec. 5.1.3) is presented in Table 5.3 with the context modeling (referring to delegation strategy (d.) in Sec. 5.1.3) based presented in Table 5.4.

In all the three tables, the randomness-based delegation model is served as the baseline. With the baseline, we can further compare it with the CTR calculated from various delegation models. The percentage in the tables shows the CTR change of each delegation model comparing to the *Random* baseline. Recommender with the best performance in each domain is highlighted in Table 5.2. Similarly, delegation model with approximately similar or better performance in each domain than those highlighted in Table 5.2 are also highlighted in Table 5.3 and Table 5.4.

As mentioned in Sec. 5.1.3, two different batch sizes are used: one with $100K$ line messages and the other with $300K$ line messages, referring to *b1* and *b3* in Table 5.3 and Table 5.4. The names of the context pre-filtering based delegation models are formatted as *algorithm_hyperparamter value_batch size* in Table 5.3. Similarly, the name context modeling based delegation models formatted as *algorithm_hyperparamter value_context used_batch size* in Table 5.4, in which *d* stands for the context 'Domain' and *i* stands for 'User-domain Impression'

Table 5.2: NewsREEL Replay Evaluation Results: Single Recommender Algorithm Baselines

Algorithm	Average CTR	ksta	tagesspiegel.de	sport1.de
Random	0.645 (0%)	0.187 (0%)	0.401 (0%)	0.749 (0%)
Path	0.632 (−2.02%)	0.137 (−26.74%)	0.357 (−10.97%)	0.748 (−0.13%)
Recent	**1.213 (88.06%)**	0.224 (19.79%)	**0.478 (19.20%)**	**1.487 (98.53%)**
Click	0.256 (−60.31%)	**0.264 (41.18%)**	0.419 (4.49%)	0.220 (−70.63%)
Imp	0.618 (−4.19%)	0.114 (−39.04%)	0.330 (−17.71%)	0.738 (−1.74%)
PRCate	0.064 (−90.08%)	0.159 (−14.97%)	0.294 (−26.68%)	0.03 (−99.60%)

5.1.6. Response Time

To ensure the delegation model to be able to run online and serve the recommendation requests in real-time, we also evaluate its performance regarding the response time. Response time distributions from the six baseline recommenders are shown in Figure 5.2. In addition, Figure 5.3 shows the response time distribution based on the contextual bandit algorithm used (*epsilon-greedy*, *UCB1*, *Thompson Sampling* and *LinUCB*). For each contextual bandit algorithm, we presents the delegation model with the highest average CTR.

Table 5.3: NewsREEL Replay Evaluation Results: Context Pre-filtering

Algorithm	Average CTR	ksta	tagesspiegel.de	sport1.de
random	0.645 (0%)	0.187 (0%)	0.401 (0%)	0.749 (0%)
epsilon_0.5_b3	0.954 (47.91%)	0.239 (27.81%)	0.437 (8.98%)	1.148 (53.27%)
epsilon_0.3_b3	1.070 (65.89%)	0.252 (34.75%)	0.468 (16.71%)	1.294 (72.76%)
epsilon_0.1_b3	1.163 (80.31%)	0.235 (25.67%)	0.456 (13.72%)	1.423 (89.99%)
ucb1_b3	1.197 (85.58%)	0.200 (6.95%)	0.420 (4.74%)	1.474 (96.80%)
ts_b3	**1.213 (88.06%)**	0.224 (19.79%)	**0.480 (19.70%)**	**1.486 (98.40%)**
epsilon_0.5_b1	0.958 (48.53%)	0.232 (24.06%)	0.449 (11.97%)	1.151 (53.67%)
epsilon_0.3_b1	1.060 (64.34%)	0.249 (33.16%)	0.458 (14.21%)	1.283 (71.30%)
epsilon_0.1_b1	1.160 (79.84%)	**0.266 (42.25%)**	0.456 (13.72%)	1.415 (88.92%)
ucb1_b1	1.195 (85.27%)	0.199 (6.42%)	0.405 (1.00%)	1.475 (96.93%)

[1] Algorithm labels: *epsilon_0.5_b3:* epsilon_greedy with $\epsilon = 0.5$ with context *Domain* and batch size $300K$, *ucb1_b3:* UCB1 with context *Domain* and batch size $300K$, *ts_b3:* Thompson Sampling with context *Domain* and batch size $300K$

Table 5.4: NewsREEL Replay Evaluation Results: Context-Modeling

Algorithm	Average CTR	ksta	tagesspiegel.de	sport1.de
random	0.645 (0%)	0.187 (0%)	0.401 (0%)	0.749 (0%)
linucb_1_d_b3	1.202 (86.36%)	0.220 (17.65%)	0.403 (0.50%)	**1.483 (98.00%)**
linucb_0.5_d_b3	1.207 (87.13%)	0.246 (31.55%)	0.423 (5.49%)	**1.483 (98.00%)**
linucb_0.2_d_b3	1.204 (86.67%)	0.257 (37.43%)	0.414 (3.24%)	**1.484 (98.13%)**
linucb_1_di_b3	1.211 (87.75%)	0.259 (38.50%)	0.451 (12.47%)	**1.485 (98.26%)**
linucb_0.5_di_b3	1.206 (86.98%)	**0.279 (49.20%)**	0.415 (3.49%)	**1.485 (98.26%)**
linucb_0.2_di_b3	**1.214 (88.22%)**	**0.268 (43.32%)**	0.467 (16.46%)	**1.485 (98.26%)**
linucb_1_d_b1	0.992 (53.80%)	0.244 (30.48%)	0.452 (12.72%)	1.180 (57.54%)
linucb_0.5_d_b1	1.205 (86.82%)	0.231 (23.53%)	0.457 (13.97%)	1.478 (97.33%)
linucb_0.2_d_b1	1.196 (85.43%)	**0.263 (40.64%)**	0.393 (−2.00%)	1.478 (97.33%)
linucb_1_di_b1	1.196 (85.43%)	0.253 (25.29%)	0.395 (−1.50%)	1.478 (97.33%)
linucb_0.5_di_b1	1.196 (85.43%)	0.207 (10.70%)	0.426 (6.23%)	1.477 (97.20%)
linucb_0.2_di_b1	1.197 (85.58%)	0.150 (−19.79%)	0.456 (13.72%)	1.479 (97.46%)

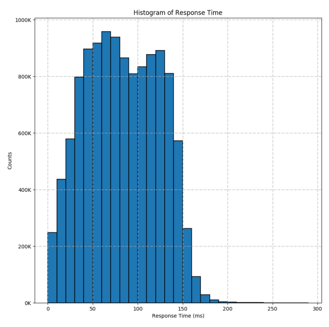
[1] Algorithm annotation: *linucb_1_d_b3:* LinUCB with $\beta = 1$ with context *Domain* and batch size $300K$, *linucb_1_di_b3:* LinUCB with $\beta = 1$ with context *Domain*, *User-Domain Impressions* and batch size $300K$

5.1.7. Catalog Coverage

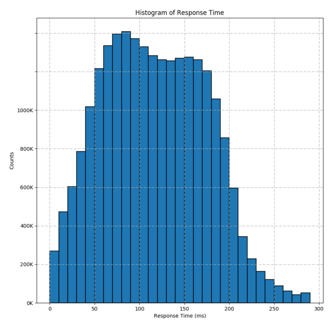
Another evaluation metric we consider is catalog coverage. Table 5.5, Table 5.6 and Table 5.7 shows the catalog coverage from the five news recommenders, context pre-filtering delegation model and context modeling delegation model respectively. With this metric, we measure the portion of distinctive items recommended to the total number of distinctive items with which users interact, see Eq. 2.4 for details.

Table 5.5: Coverage: Single Recommender Algorithm

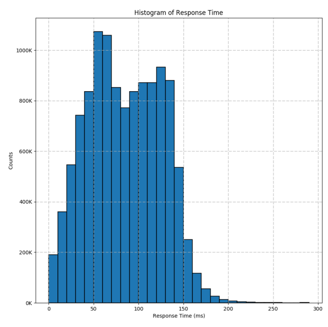
Algorithm	Average CTR	ksta	tagesspiegel.de	sport1.de
Random	86.81%	97.77%	87.13%	91.83%
Path	39.55%	42.53%	12.17%	38.61%
Recent	99.99%	99.99%	99.97%	99.99%
Click	0.53%	0.55%	4.51%	0.87%
Imp	41.03%	44.34%	11.27%	40.04%
PRCate	4.57%	3.26%	0.35%	3.62%



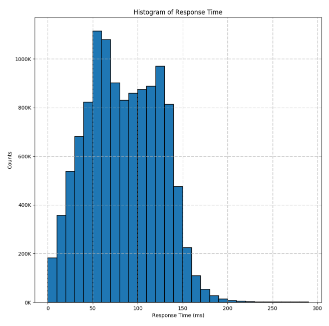
(a) Response Time: Random Recommender



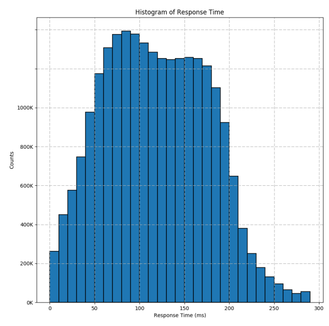
(b) Response Time: Path Recommender



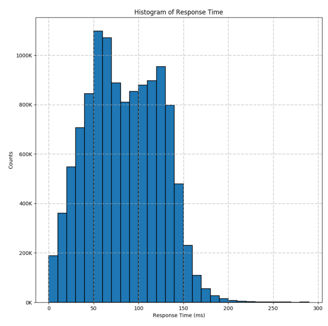
(c) Response Time: Recent Recommender



(d) Response Time: Click Recommender



(e) Response Time: Imp Recommender



(f) Response Time: PRCate Recommender

Figure 5.2: Baseline Recommender: Response Time

Table 5.6: Coverage: Context Pre-filtering

Algorithm	ksta	tagesspiegel.de	sport1.de	Overall Coverage
epsilon_0.5_b3	69.85%	99.53%	99.15%	85.84%
epsilon_0.3_b3	56.51%	99.55%	99.29%	79.72%
epsilon_0.1_b3	62.33%	94.42%	99.39%	80.08%
ucb1_b3	39.55%	45.45%	97.85%	47.14%
ts_b3	99.77%	99.86%	99.52%	99.79%
epsilon_0.5_b1	75.10%	99.77%	99.12%	88.36%
epsilon_0.3_b1	57.12%	97.23%	99.37%	78.95%
epsilon_0.1_b1	33.34%	89.77%	99.49%	64.62%
ucb1_b1	39.78%	45.93%	98.40%	47.51%

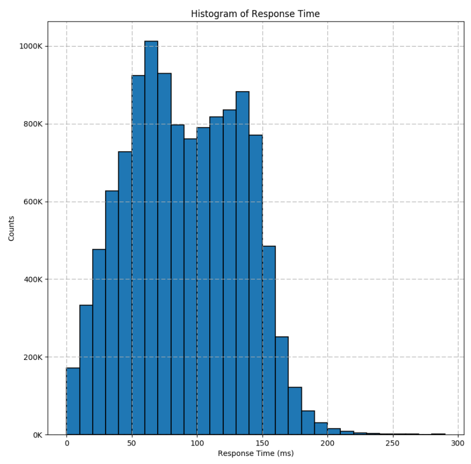
5.2. NewsREEL Live Online Evaluation

5.2.1. Online Evaluation Structure

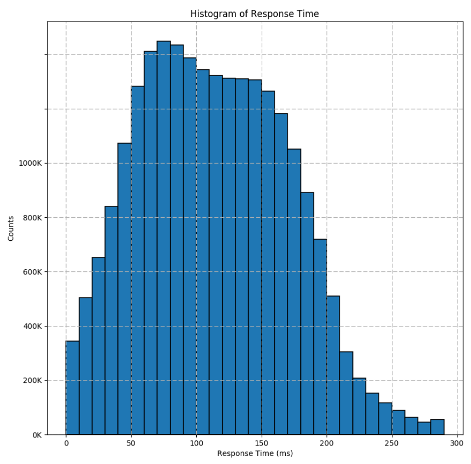
The Open Recommendation Platform (ORP) is responsible for redirecting the real-world traffic from publishers to participants and monitoring the performance of their deployed recommenders. With ORP, participants are able to deploy their recommender services, receive recommendation requests and give back recommendations.

5.2.2. Online Method

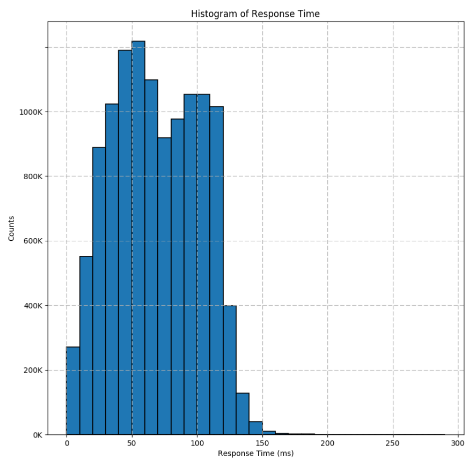
The NewsREEL Live online evaluation took place from 24 April to 7 May 2017. A daily data stream redirected to each participant in the Live task consists of only a small portion Plista daily data. Due to the low data volume, our delegation model cannot be trained to update in the online scenario. The low



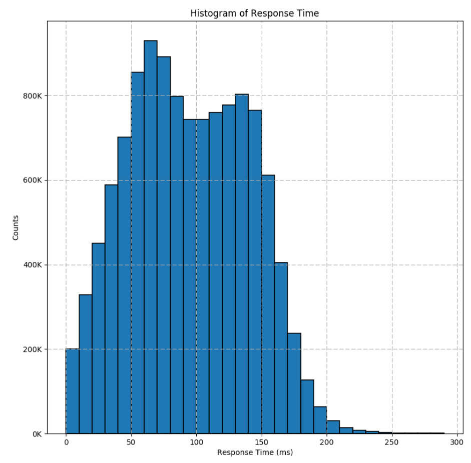
(a) Response Time: Delegation Model based on epsilon-greedy with $\epsilon = 0.1$ and batch size $300K$



(b) Response Time: Delegation Model based on UCB1 with batch size $300K$



(c) Response Time: Delegation Model based on Thompson Sampling with batch size $300K$



(d) Response Time: Delegation Model based on epsilon-greedy with $\beta = 0.2$, context *Domain*, *User-Domain Impressions* and batch size $300K$

Figure 5.3: Delegation Model based Recommender: Response Time

volume of data refers to the small number of recommendation requests, user clicks and item updates.

As the result, we deployed the delegation model with the best performance trained from the Replay task, referring to delegation model based on LinUCB with hyper-parameter $\beta = 0.2$, context *Domain* and *User-Domain Impressions* and batch size $300K$ (*linucb_0.2_di_b3* in Table 5.4). Another challenge of the online evaluation is that a large amount of traffic comes from two domains, *bz-berlin.de (17614)* and *motor-talk.de (13554)*. Domain *bz-berlin.de (17614)* is a new domain which is not included in the replay dataset, while domain *13554* is included in the Replay dataset but not in the offline training due to its low CTR performance (See Sec. 5.3.2 for detail). The trained delegation model knows nothing about these two domains. To solve the problem, we delegate all the recommendation requests from those two domains to a default recommender *PRCate*. The challenges might lower our recommendation performance in the Live task. In total of the 19 teams that participated in the Live task, our delegation model ranked 10th and slightly outperforms the baseline.

Table 5.7: Coverage: Context Modeling

Algorithm	ksta	tagesspiegel.de	sport1.de	Overall Coverage
linucb_1_d_b3	34.01%	46.14%	99.47%	45.04%
linucb_0.5_d_b3	18.99%	43.34%	99.39%	36.84%
linucb_0.2_d_b3	7.59%	14.89%	99.58%	18.64%
linucb_1_di_b3	50.13%	32.93%	99.50%	46.44%
linucb_0.5_di_b3	**55.73%**	32.71%	99.37%	48.90%
linucb_0.2_di_b3	**62.80%**	71.71%	99.41%	69.94%
linucb_1_d_b1	34.74%	45.94%	99.39%	45.28%
linucb_0.5_d_b1	30.68%	94.19%	99.47%	65.41%
linucb_0.2_d_b1	28.06%	44.13%	99.33%	41.37%
linucb_1_di_b1	21.89%	34.36%	99.47%	34.09%
linucb_0.5_di_b1	47.67%	44.32%	99.44%	50.49%
linucb_0.2_di_b1	23.68%	96.62%	99.58%	63.30%

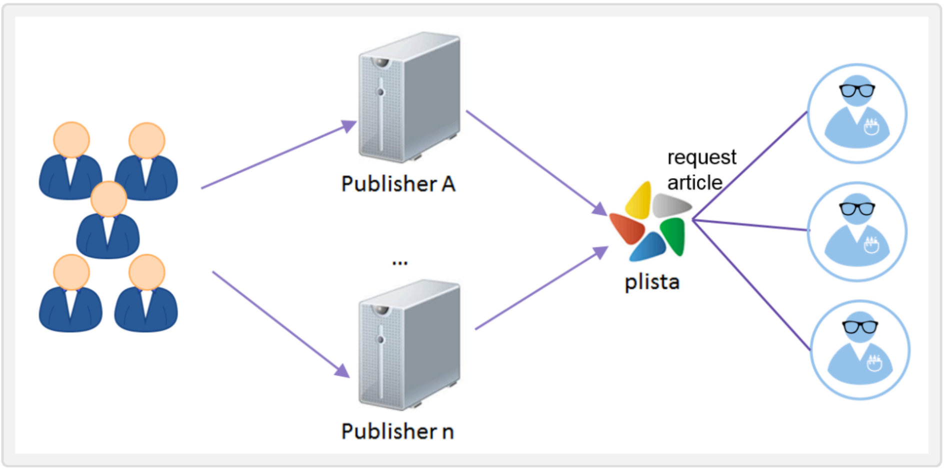


Figure 5.4: Online Evaluation Structure, source from [42]

5.2.3. Online Click Through Rate

Table 5.8 shows the online evaluation results of our delegation model regarding the *CTR*. In the online scenario, publishers operate news portals where users can access news articles. A recommendation request is triggered whenever a user lands on a news articles page with real-time recommendations loaded by widgets on that page. On receiving the recommendation requests, Plista will delegate a portion of recommendation request the Open Recommendation Platform (ORP, see Sec. 4.1). Then ORP will further forward the requests to all participants, and upon receiving the recommendation responses from the participants, it will randomly select a list of recommendations from all valid responses to return and display in the widget areas. Two types of *impressions* are created by ORP. The first type *widget impressions* considers user's clicks on the widget while the second type *item impressions* considers user's clicks on the single news articles. For this reason, the total number of *item impressions* is N times more than the *widget impressions*, in which N is the total number of news articles required in the recommendation, with the number depending on the size and design of widget). With the two different types of impressions, there are also two different types of CTR. The metric for the recommendation performance is measured by the online Click Through Rate as described in Eq. 2.3, which equals to the number of clicks divided by the number of impressions.

5.3. Limitations

5.3.1. Time Constraint

Only a subset of the Replay dataset is used for the near-online evaluation. The amount of data is not enough comparing to other work in NewsREEL Challenge. The reason is due to the time spent on the

Table 5.8: NewsREEL Live Evaluation Results from 24 April to 7 May 2017 (delegation model: *linucb_0.2_di_b3* trained from offline)

Metric	Click Num	Impression Num	CTR
Widget	747	60, 814	0.0123
Individual item	747	192, 207	0.0039
Baseline widget	726	62, 052	0.0117
Baseline individual item	726	193, 014	0.0038

replay. The replay of a daily dump takes about 2 hours with a month worth of data (28 days) 56 hours. For the replay task, we test 28 different delegation models, it would take 63 days if the model is tested with a month of data. In the replay task, only a three-day subset is used for training the delegation model. However, the subset contains about 20 million messages, which is of the same magnitude as the dataset used in work [14] and much larger than the data redirected to our recommender during the Live Task.

5.3.2. Dataset

In this section, we consider the limitations given by the offline dataset and online streaming data. As indicated in Table 5.1, the offline dataset is not evenly distributed, about 70% of the traffic is from domain *sport1.de*. As the result, average CTR is greatly influenced by the CTR from the domain. Domain dominance as well as the overwhelming performance of the *Recent* recommender in that domain makes it hard for the delegation model to improve the average CTR. In addition, the volume of the online stream data is too small to train and update our delegation model. The total number of recommendation requests we received during the online evaluation period takes only about 1% of the three-day dataset used for the offline evaluation. For this reason, the delegation model implemented in the online evaluation is trained by the offline dataset. However, for some domain, like *motor-talk.de (13554)*, we are not able to learn from the offline dataset since the amount of data from that domain in the offline dataset is too low. Furthermore, this year we noticed a new publisher *bz-berlin.de (17614)*, which is not included in the offline dataset, appears in the online evaluation with a large amount of traffic received online from that domain. To deal with the recommendation requests from these domains, we set a fallback recommender in the online evaluation. We believe the online data challenges can to some extent decrease the online recommendation performance regarding the CTR.

6 Discussion and Future Work

In this chapter, we discuss our findings, draw conclusions regarding the research questions and suggest future work.

6.1. NewsREEL Replay

In the replay task, two evaluation aspects are taken into consideration: (1) CTR, (2) Response time and (3) Catalog coverage.

6.1.1. CTR Performance

To explain the results, we first illustrate two facts with respect to the domain *sport1.de*. Firstly, as explained in the previous section, about $70\% - 75\%$ of the traffic comes from this domain. Thus, the CTR increase from this domain leads to an increase of the average CTR. In addition, the *Recent* recommender shows an overwhelming performance compared to any other recommender in this domain. The dataset imbalance and the overwhelming performance of the *Recent* recommender in domain *sport1.de* caused a failure to our initial delegation model. At first, we built a delegation model regardless of any context. During the training process, the delegation model receives a large amount of positive feedback ('click' events) from the *Recent* recommender. As the result, the delegation model chooses the *Recent* recommender to serve the recommendation requests for most of its time. The delegation model is affected by the recommender performance in domain *sport1.de* from which a large amount of traffic comes. The 'best' recommender from its experience is the 'best' recommender in domain *sport1.de*. Furthermore, due to the overwhelming performance of the *Recent* recommender in *sport1.de*, it seems the delegation model does not benefit the CTR performance when compared with the *Recent* recommender and even causes a CTR decrease in that domain. Further exploration in domain *sport1.de* is less useful.

When looking at the CTR performance of the delegation model, we can see that all delegation models outperform the random baseline. The result suggests the delegation models can learn from the past and choose appropriate recommenders for the incoming recommendation requests to improve the CTR performance. Besides, the performance of the delegation model varies when using different contextual algorithms, hyper-parameter values and batch sizes.

The context pre-filtering model is tested with different hyper-parameter values and batch sizes. In epsilon-greedy, the fixed parameter ϵ is served to balance the exploration and exploitation. Generally, smaller ϵ values show better results. From the side of epsilon-greedy, exploring too much is not a good idea. Usually, ϵ value is set below 0.5 in real-world settings to prevent too much exploration. UCB1 works better than epsilon-greedy in domain *sport1.de*, but worse in domain *ksta* and *tagesspiegel.de*. The result is a little bit unexpected since UCB1 explores under the guidance of a confidence bound [14], while epsilon-greedy explores randomly without any guidance. The reason is probably due to the fact that the parameter is used to balance the exploration and exploitation is fixed in UCB1. UCB with lower ρ value (explore less and exploit more) would probably benefit the CTR performance [28]. The advanced algorithm, Thompson Sampling, works well in the evaluation, and it even shows the best performance among all other algorithms in domain *tagesspiegel.de*. The batch size does not seem to influence the performance of the delegation model too much.

The context modeling delegation model generally shows better CTR performance in domain *sport1.de* and *tagesspiegel.de* than the context pre-filtering delegation model. A better overall performance is achieved by the context modeling delegation model. With respect to batch size, delegation models with larger batch size and similar hyper-parameter tends to perform better except for several cases in domain *tagesspiegel.de*. When batch size is set to $300K$, the delegation model with smaller hyper-parameter value is more likely to show higher performance. Furthermore, the set of delegation models with two different types of context and batch size $300K$ are among the ones with top overall performance. Even though adding one more context does not always help improve the CTR performance, the delegation model with the highest CTR performance is built by incorporating the two context: *Domain* and *User-domain Impression*. The model even slightly outperforms the strong *Recent* baseline regarding its average CTR. As shown in Table 5.4, *linucb_0.2_di_b3* outperforms the *Recent* recommender in Domain *ksta*, but performs slightly worse in domain *tagesspiegel.de* and domain *sport1.de*. However, when it comes to the average overall CTR, the superior performance of the delegation model makes it able to even slightly beat the *Recent* model.

In summary, it is important to make a trade-off between exploration and exploitation when building a delegation model based on contextual bandit algorithm. To do so, the hyper-parameter value used to balance the exploration and exploitation should be taken into consideration. In addition, batch size sometimes can influence the CTR performance of the delegation model. Tunning the two value would help to build a better delegation model. Furthermore, incorporating more context can also help to build a better delegation model. Also, as indicated in the previous study, the recommender's performance depends heavily on the domain. From the side of different domain, further exploration in domain *sport1.de* tends to decrease its corresponding CTR performance, but the exploration does increase the CTR in domain *ksta* and further improve the overall CTR performance.

6.1.2. Response Time

In the offline evaluation, we measure the response time distribution of the baseline recommenders and our proposed delegation models. As shown from Figure 5.2, most of the recommendation responses are sent back within 200ms with baseline recommenders. In addition, the figure shows that response time varies among different recommenders. For example, both *Path* and *Imp* recommenders show a slightly higher response time. The difference also impacts the recommendation response time of the delegation model.

The response time of the delegation model varies based on its delegation strategy, namely which recommender to choose to serve the incoming recommendation requests. However, as indicated in Sec. 3.2.1, where we show there is no time waste when delegating recommendation requests (the delegation strategy in each batch is built beforehand), the delegation models are guaranteed to give valid recommendations no later than the recommender with the highest response time. In other words, the upper bound of the response time is determined by the recommender with the highest response time. Furthermore, in both figures, response time sometimes exceeds the 100ms time limit required by the online scenario. This is because in the offline scenario, up to 1000 recommendation requests are sent at the same time with multi-thread to save the replay time, while in the online scenario, the number of recommendation requests we received is much smaller than the number we received offline.

6.1.3. Catalog Coverage

In the offline evaluation, we also measure the catalog coverage of our recommendation as defined in Sec. 2.4.2. Table 5.5 shows the catalog coverage of the five recommenders. We could see the recommender *Recent* has the highest coverage up to 99% in all the three domains. Each time a recommendation request comes, it would recommend the most recent requested articles to the current user. Basically, it would recommend all news articles that users have interacted with and thus lead to a very high coverage. Moreover, catalog coverage of the four popularity-based recommenders are much lower, especially the catalog coverage of the *Click* recommender and *PRCate* recommender. For the *Click* recommender, news articles to be recommended are from a small item set which only contains the 'clicked' item. The coverage of the *PRCate* recommender is low because only news articles in the same category as the current article can be recommended.

When turning to Table 5.6 and Table 5.7, we find regardless of the contextual bandit algorithms used, catalog coverage from the domain *sport1.de* remains high all the time. The finding is consistent with our finding in CTR. The recommender *Recent* shows an overwhelming CTR performance in the domain, and our delegation models are able to learn the fact. The more the delegation model relies on the recommender *Recent*, the higher the coverage is. Furthermore, the catalog coverage can also give us some clues about which recommender algorithms are chosen during the recommendation. High coverage is a result of having the recommender *Recent* to give recommendation most of the time. Last but not least, recommender algorithms with low coverage, like the *Click* recommender, are compensated by recommender algorithms with high coverage when they are combined to provide recommendations together. For example, for the two delegation models which are based on *linucb_0.2_di_b3* and *linucb_0.5_di_b3* respectively and reach the highest CTR in domain *ksta*, they also show a much higher coverage when comparing to the single baseline recommender *Click*, of which the coverage is only 4.51%. The two delegation models at least benefit from the recommenders, the *Click* recommender with higher CTR and the *Recent* recommender with higher coverage.

6.2. NewsREEL Live

As described in Sec. 5.2.2, the online delegation model is trained from the offline dataset which does not cover the whole online dataset and is not able to update online. These challenges might lower the CTR performance of the online performance of our proposed delegation-model based recommender. It still slightly outperforms the baseline. In addition, it is able to offer recommendation responses within the 100ms time constraint.

6.3. Discussion

The delegation model is scalable regarding adding more context features and news recommenders. With the context modeling method, context features can easily be encoded to the delegation model. In addition, the computational complexity of Linear UCB is linear in the number of arms (namely news recommenders in our case) and at most cubic in the number of features [14]. The low computational complexity makes the delegation model easy to update. With the batch update scheme, the delegation model can also effectively deal with the delayed feedback from users in real-world settings. In one word, the delegation model is feasible to run in the real-word settings. Furthermore, we believe that the delegation model would work better if we can utilize both with the near-online evaluation and online evaluation. Firstly, we assume the offline dataset is consistent with the data stream from the online scenario and the near-online evaluation is consistent with the online evaluation. Then for the next step, various delegation models with different hyper-parameters and batch sizes can be trained in a near-online scenario. The model with the highest performance is then implemented online. In the online scenario, only a part of traffic redirected to the delegation model is used to continuously train and update it in batches after receiving some 'click' events (here, we assume we can track the 'click' events), and the rest traffic will make use of the delegation without doing anything about the updating.

6.4. Conclusion

To conclude, let's recap our research questions:

- Is there a way in which the delegation strategy is able to update continuously based on users' feedback so that it can choose the appropriate recommender to serve recommendations during the online news recommendation?

  (Can a Multi-armed Bandit algorithm be a way to continuously update the delegation strategy from the click behaviors learned from users during the online news recommendation?)

  - Can a Multi-armed Bandit algorithm combined with context help to build a better delegation strategy in the online news recommendation? Furthermore, what kind of context can be exploited and how the context is combined?
  - Is it possible for the proposed delegation model to run online in order to serve the recommendation requests in the online news recommendation scenario? In other words, can the delegation model feasible to run online to provide valid recommendation requests within a specified time limit?

6.4.1. Main Research Questions

In this section, we discuss our main research question:

- Is there a way in which the delegation strategy is able to update continuously based on users' feedback so that it can choose the appropriate recommender to serve recommendations during the online news recommendation?

  (Can a Multi-armed Bandit algorithm be a way to continuously update the delegation strategy from the click behaviors learned from users during the online news recommendation?)

In the near-online evaluation, we have demonstrated that the delegation model based contextual bandit algorithms are able to update continuously with users' feedback and choose appropriate recommender for the incoming recommendation requests with the purpose to maximize the overall CTR.

6.4.2. First Sub-question

Furthermore, we discuss our first sub-question:

- Can a Multi-armed Bandit algorithm combined with context help to build a better delegation strategy in the online news recommendation? Furthermore, what kind of context can be exploited and how the context is combined?

Incorporating more context can also help to build a better delegation model. Here, we consider two types of context *Domain* and *User-Domain Impressions*, which are extracted directly from the incoming recommendation requests. Two different types of delegation model, delegation model based on context pre-filtering and delegation model based on context modeling, are proposed based on how context is combined with the bandit algorithms.

6.4.3. Second Sub-question

Here, we discuss our second sub-question:

- Is it possible for the proposed delegation model to run online in order to serve the recommendation requests in the online news recommendation scenario? In other words, can the delegation model feasible to run online to provide valid recommendation requests within a specified time limit?

Although our proposed delegation model is only trained and updated in a near-online scenario, it is also feasible to update continuously online if enough traffic is redirected to our delegation model. In the near-online evaluation, we also evaluate the response time with the concern that recommendation response must be given in a time limit in real-world settings. Most of the recommendation responses can be given within 200ms. The response time is slightly longer than the 100ms required by the online scenario due to the large number of recommendation requests sent at the same time with multi-thread.

6.4.4. Catalog Coverage

We evaluate the recommendation performance of the proposed delegation model with another evaluation metric, catalog coverage. Popularity-based recommender algorithms generally have a lower coverage since they always focus on a small number of popular news articles. However, when different recommender algorithms are combined to provide recommendations, the delegation model can sometimes benefit from both recommenders with higher CTR but slightly lower coverage and recommenders with higher coverage but lower CTR.

6.5. Future Work

In this section, we discuss possible future work with respect to four aspects: (1) exploring other context features, (2) combining more recommenders, (3) reward definition, and (4) evaluation

6.5.1. Exploring other Context Features

Apart from the context features that are extracted from the recommendation requests directly, we can also construct new context features based on the features extracted from the incoming recommendation requests. For example, combine context features with temporal information, e.g. user-domain impressions in a time window. For example, if a user visits the news portal several times in a short time after the occurrence of some popular event, a popularity-based recommender would work well under that situation. However, if a user visits the news portal several times in the morning with nothing special happening in the news field, he/she would more likely to show a personal preference towards the news articles. At that point, a personalized recommendation is probably more useful. In the above examples, time of the day, e.g. morning, the occurrence of popular events and the user-item impressions in a time window can all be considered as context. In that case, the delegation model is more complex to build since the extracted context feature cannot be used directly and it is also hard to predict the happening of the popular events in advance. Table 6.1 shows a list of possible useful context features.

Table 6.1: Useful Context Features

Context	Description
Time Weekday	Days of the week
Time Hour	Hours of the day
User Domain (publisher) Impression	How many times the user interacts with the domain
Device type	Device the user used, can be desktop, phone, and etc.
Domain	Current news portal from where the message comes from
Happening of Weather Problem	Whether there is a weather problem in the system
User Domain Impression (time window)	User domain impressions in a time window
Item Popularity	Popularity of the item that the user is currently reading

6.5.2. Combining More Recommender

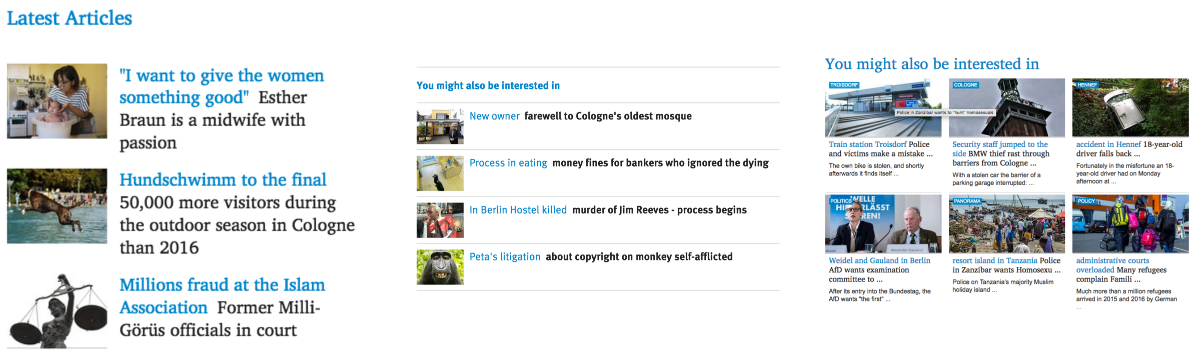
Our delegation model is built on top of four popularity-based recommender algorithms and one recency-based recommender algorithms. The recommendation performance of our delegation model is limited by the underlying recommender algorithms. In our future work, we would like to add more recommenders, especially those showing good performance [44] in this year's CLEF-NewsREEL Challenge in order to further improve the CTR. Further, we would also like to add some personalized recommendation method, such as item-based collaborative filtering in which only $50,000$ most recent user-item interactions are kept for providing recommendations [7].

6.5.3. Reward Definition

In the project, our main concern is CTR and we define the corresponding reward in the contextual bandit algorithm as clicks received from the users. By defining rewards as clicks from users, the delegation model, which is built from various contextual bandit algorithms, can choose the appropriate recommender for the recommendation requests with the purpose to maximize the total sum of clicks (CTR). However, in real-world, rewards can also be defined as revenues from the clicks. Clicks on different news articles may trigger different amounts of revenues, which is often the case in the advertisement field. With this thought, rewards can also be defined as revenues from the clicks to maximize the final profit earned during the recommendation in a business model. Furthermore, to optimize various aspects of the recommendation based on the evaluation metric, e.g., catalog coverage, the contextual bandit algorithms can reward the recommender, which increases the catalog coverage during the recommendation. Rewards can then be further defined as a weighted combination of clicks and other observations regarding what we would like to improve. In summary, rewards can be defined as many different forms regarding what we would like to maximize during the recommendation.

6.5.4. Evaluation

It has been suggested in the previous work that news page layout can affect the evaluation of news recommender [24]. In order to avoid the influence of page layout, some previous work [14, 24] only considers users' clicks on the news articles in the specific prominent positions. However, in the online evaluation, we have few clues about in what way the recommended articles will be displayed. Figure 6.1 shows three different recommendation widgets of a web page from the news portal *ksta.de*. In addition, CTR performance evaluated from the near-online scenario is to some extent influenced by the real recommendations given during the recording time. In other words, the near-online evaluation



Latest Articles

"I want to give the women something good" Esther Braun is a midwife with passion

Hundschwimm to the final 50,000 more visitors during the outdoor season in Cologne than 2016

Millions fraud at the Islam Association Former Milli-Görüs officials in court

You might also be interested in

New owner farewell to Cologne's oldest mosque

Process in eating money fines for bankers who ignored the dying

In Berlin Hostel killed murder of Jim Reeves - process begins

Peta's litigation about copyright on monkey self-afflicted

You might also be interested in

TROISDORF Train station Troisdorf Police and victims make a mistake ... The own bike is stolen, and shortly afterwards it finds itself ...

Police in Zanzibar wants to "hunt" homosexuals

COLOGNE Security staff jumped to the side BMW thief rast through barriers from Cologne ... With a stolen car the barrier of a parking garage interrupted: ...

HENNEF accident in Hennef 18-year-old driver falls back ... Fortunately in the misfortune an 18-year-old driver had on Monday afternoon at ...

POLITICS Weidel and Gauland in Berlin AfD wants examination committee to ... After its entry into the Bundestag, the AfD wants "the first" ...

PANORAMA resort island in Tanzania Police in Zanzibar wants Homosexu ... Police on Tanzania's majority Muslim holiday island ...

POLICY administrative courts overloaded Many refugees complain Famili ... Much more than a million refugees arrived in 2015 and 2016 by German ...

Figure 6.1: Page Layout Example

performance of our recommender is affected by the recommender used by Plista since users are not able to click on the recommendations in the scenario and their behaviors are influenced by the news articles shown to them during the recording time. Besides, it has been suggested that the recommender's performance in the near-online scenario is not consistent with its performance in the online scenario [45].

In the thesis, we also tried another evaluation metric Catalog Coverage in order to measure the catalog coverage of the news article. But in real life, CTR still seems to be the most suitable evaluation metric for news recommendation. Although people sometimes would expect a diversity of news articles, they focus on the popular news articles most of the times. Furthermore, even if they are not interested in the popular news articles, the popular news articles shown on the websites would increase users' trust to the system. Furthermore, users may click on the recommended news articles due to the clickbait headlines or images, but later find he/she is not interested in the recommendation. One possible way to tackle the problem is to consider the time spent by the user on the clicked webpage.

A

Appendix A: Paper

This appendix contains the paper written during the thesis and published in Conference and Lab Evaluation Forum (CLEF) 2017. The paper is now accessible online [46].

CLEF NewsREEL 2017: Contextual Bandit News Recommendation

Yu Liang[1], Babak Loni[1] and Martha Larson[1,2]

[1] Delft University of Technology, Netherlands
[2] Radboud University Nijmegen, Netherlands
Y.Liang-2@student.tudelft.nl, b.loni@tudelft.nl, m.a.larson@tudelft.nl

Abstract. In the CLEF NewsREEL 2017 challenge, we build a delegation model based on the contextual bandit algorithm. Our goal is to investigate whether a bandit approach combined with context extracted from the user side, from the item side and from user-item interaction can help choose the appropriate recommender from a recommender algorithm pool for the incoming recommendation requests. We took part in both tasks: NewsREEL Live and NewsREEL Replay. In the experiment, we test several bandit approaches with two types of context features. The result from NewsREEL Replay suggests that delegation model based on the contextual bandit algorithm can improve the click through rate (CTR). In NewsREEL Live, a similar delegation model is implemented. However, the delegation model from NewsREEL Live is trained by the data stream from NewsREEL Replay. This is due to the fact that the low volume of data received from the online scenario is not enough to support the training of the delegation model. For our future work, we will add more recommender algorithms to the recommender algorithm pool and explores other context features.

Keywords: Recommender System, Context, Contextual Bandit, News, Evaluation

1 Introduction

The CLEF NewsREEL Challenge [1, 2] is a challenge in which participants are asked to provide effective real-time news recommendation. The challenge provides participants with a platform to evaluate their recommender algorithms with millions of real-world users. We took part in both tasks [3]: Task 1 (NewsREEL Live) [4] and Task 2 (NewsREEL Replay) [5]. In the Live task, the recommendation requests are distributed to all participants of the challenge and thus not all traffic is sent to a single participant. This makes it more challenging than the Replay task. In the Live task, we use a trained delegation model that is implemented for the Replay task.

Our work was initially inspired by Lommatzsch and Albayrak, who, in their work [6], combine several recommender algorithms with one delegation model responsible for delegating the incoming recommendation request to the appropriate recommender algorithm. In our work, we also build a similar delegation model with the objective to improve recommendation performance, in other words, to maximize Click Through Rate (CTR). We find that a multi-armed bandit (MAB) algorithm is suited for the purpose. Multi-armed bandit algorithms address the problem of a gambler at a row of slot machines who has to decide which machine to play, in what sequence to play and how many times to play in order to maximize the rewards collecting from the slot machines. The model has been widely used in website optimization [7] for choosing the right ads for users [8] and selecting the right advertisement format [9]. All of these models involve optimization problems, e.g., selecting the suitable ad format in order to maximize the CTR received from users. Another good property of MAB algorithms is that they can continuously learn from the past, such as from user feedback, and then adjust its decisions about which arm to choose. With the feedback, it is able to balance the exploration and exploitation dilemma. Exploitation means the MAB algorithm chooses the arm with the currently best performance from the previous plays, while exploration means it chooses the arm with lower performance in order to learn more from these arms. This kind of trade-off is missing in A/B testing, in which a pure exploration is applied at the initial phase followed by a pure exploitation in a very long period [7]. The exploration and exploitation trade-off makes MAB algorithm works well for the news recommendation [10], in which it is able to recommend popular items most of the time and also explore the news items in the long tail. Further, combined with context, the model works even better. Context is proved to be useful for the optimization [7–10].

However, our consideration is not directly related to news items, but the recommender algorithms. Given the effective recommender algorithms from previous years, we believe the contextual bandit algorithm is able to combine them in a wise way and ultimately improve the recommendation performance. Our research interest comes with the assumption that given the context of a user, item and user-item interaction, contextual bandit algorithms can help choose appropriate recommender algorithm for each recommendation request and thereby improve the CTR.

This paper is structured as follows: in Section 2, we discuss the work related to the contextual multi-armed bandit algorithm. In Section 3, we introduce the method that we implemented in this work. Section 4 describes our evaluation methods, including NewsREEL Replay and NewsREEL Live. In Section 5, we analyze the evaluation results. In the last section, we summarize the results and discuss future work.

2 Background and Related Work

2.1 Multi-armed Bandit Algorithms

epsilon-Greedy Epsilon-Greedy is the simplest algorithm to solve the multi-armed bandit problem. The only parameter in the model is ϵ. In the standard settings, ϵ is fixed. With probability $1 - \epsilon$, the policy plays the currently best arm, and with probability ϵ, it plays a random arm. The algorithms also have several variations, e.g., ϵ-decreasing in which ϵ decreases over time [11].

Upper Confidence Bound Unlike epsilon-greedy, UCB is not only concerned about the reward from the previous plays, but also about how much it knows about the available arms. At the initial phase, UCB explores each arm at least once. After the initial phase, each time it plays the arm with the highest value from the formula $r_\alpha + b$, where r_α is the estimated reward for arm α, and b is a special bonus for the arm α. One widely used UCB variation is UCB1 [12], in which b is set to be$\sqrt{\frac{2ln \sum_{i=1}^{n} t_i}{t_\alpha}}$, t_i is the number of plays arm i has been chosen, and t_α is the number of plays arm α has been chosen. In UCB, arms that are explored less in the previous plays would receive higher bonus value, while arms that are explored more would receive lower bonus value. This can prevent the under-exploration of potential rewarding arms.

Thompson Sampling Another widely used multi-armed bandit algorithm is Thompson Sampling. Thompson sampling models the probability distribution of each arm's reward. The reward of arm α follows a probability distribution with mean θ_α^* [13]. In the case where bandits are binary, a Beta-Bernoulli distribution [14] is used. The algorithms repeatedly draw a sample from the distribution of each arm, and it plays the arm with the largest value.

2.2 Contextual Bandit Algorithms

Contextual bandit algorithms combine MABs with context features. In our settings, users and news articles are always represented by feature vectors, such as domains. The contextual bandit algorithm chooses arms in such a way that the most appropriate arm for each context is played to maximize the reward. A trivial implementation is to build a MAB model for each context. Another idea is assuming there is a relationship between the expected reward and the context. We implement both ideas.

Linear UCB Linear UCB is a contextual bandit algorithm first proposed in [10]. The algorithm assumes that there is a linear relationship between the expected reward and the context. In this algorithm, the context is used to update the rewards and improve its arm selection strategy. In each trial (play) t, the expected payoff of arm α is equal to $E[r_{t,\alpha}|x_{t,\alpha}] = x_{t,\alpha}\theta_\alpha^*$, where $r_{t,\alpha}$ specifies the reward of arm α at trial t, $x_{t,\alpha}$ specifies the feature vector for arm α at trial t and θ_α^* is a coefficient that can be learned from the previous trials. In each trial, the algorithm plays the arm with the highest final score. The final score of each arm is calculated as:

$$\alpha_t = \text{argmax}_{\alpha \in \mathtt{A}} \left(x_{t,\alpha}\hat{\theta}_\alpha + \beta\sqrt{x_{t,\alpha}^T A_\alpha^{-1} x_{t,\alpha}} \right) \tag{1}$$

where $\mathtt{A}$ is the set of arms, $\hat{\theta}_\alpha$ is an estimate of coefficients, β is a hyper-parameter and A is the covariance matrix of coefficients.

3 Framework

Our framework is designed to validate the hypothesis that contextual bandit algorithms can choose the appropriate recommender algorithm from the recommender algorithm pool for the incoming recommendation requests given context from users and items. In NewsREEL Replay, we consider two types of training: the training of the delegation model and the training of the recommender algorithms. The delegation model is trained and updated in batches (a more detailed discussion is presented in Section 3.2). In contrast, the recommender algorithms are trained and are able to update continuously regardless of the batches. The delegation model implemented in the Live task is also trained in the same way, but the data stream for training is from the offline dataset due to the low volume received from the online scenario. The recommender algorithms in the Live scenario also update continuously.

3.1 Recommender Algorithms

We build up our delegation recommender based on the recommender developed by recommenders.net. The recommender algorithm pool consists of five simple recommender algorithms:

- *Recent*: Recommend the most recently created or updated articles. For the realization, a ring buffer is implemented to store the most recently created or updated articles. The size of the ring buffer is set to 100. When there is a new item, it will be inserted into the ring buffer. If the ring buffer reaches its size, it will drop the least recent news articles.
- *Path*: Given the news articles that the user is currently reading, recommend the most popular[3] news articles requested next by users, referring to the *Most Popular Sequence* in the work [6]. In the default settings, for each news article, 100 most recently requested articles are kept. The popularity scores are then computed from the 100 articles.
- *Click*: Similar to the algorithm *Path*, but the popularity algorithm considers only the clicks[4] of the user.
- *Imp*: Similar to the algorithm *Path*, but the popularity algorithm considers only the impressions from users.
- *PRCate*: Popularity and category based Recommender. Recommend the news article with most impressions and clicks within a certain category. The popularity scores only consider 100 most recently requested news articles within the category.

[3] The popularity is computed from the weighted combination of impressions and clicks.
[4] A click is different from an impression: a click means the user clicks on the recommendation, while an impression means the user clicks on articles that were not necessarily recommended [15]

3.2 Delegation Framework

Our proposed delegation framework is trained and updates only in the Replay task. The low volume of data from the Live task is not enough for training such a framework. However, in NewsREEL Live, we implement the delegation model trained from the NewsREEL Replay. More details about NewsREEL Live can be found in Section 4.1. It is not trivial to implement MAB algorithm in the real-world settings. We first define the reward. In our settings, the reward is defined as clicks received from recommendations. If a user clicks on a recommendation, the reward is 1, otherwise, the reward is 0. Therefore, the reward is a binary value. Given the context, the recommender algorithm, and its corresponding reward, the delegation model is able to update. However, unlike what is implemented in [9, 10], where rewards can be obtained immediately after each action, rewards in real-world settings are delayed. Specifically, in our case, there would be hundreds of incoming recommendation requests between a recommendation and its corresponding reward. It is unrealistic to suspend the delegation model before receiving the reward or to update the whole delegation model each time a reward is observed. We implement a similar batch update as used in the work [8] for our delegation model. The basic underlying idea is to update the delegation model in batches, in other words, the model updates only after receiving enough amount of reward.

Batch update and assignment The whole recommendation process is divided into batches. The contextual bandit algorithm is updated at the end of each batch with the rewards collected in each batch, as illustrated in Figure 1. After each update, the delegation model will preassign appropriate recommender for each context to serve incoming requests in the next batch.

Initialization In the first batch, there is no historical data from which the contextual bandit algorithm can learn the model. Therefore, at the initial phase of recommendation, recommender algorithms are randomly selected to serve the incoming recommendation requests. At the end of the first batch, the contextual bandit algorithm is able to update for the first time with the rewards collected in the first batch.

Rewards Rewards are defined as ‘clicks’ received from the recommendations. However, in both tasks, we cannot directly track whether our previous recommendations are successful or not. Making use of the evaluator from NewsREEL Replay, the reward is defined, for a given recommendation, based on whether the user interacts with the item in the next 10 minutes.

3.3 Exploiting Context

We exploit context with two different methods. In the first method, the context is used to filter the model. For example, if the recommendation request belongs

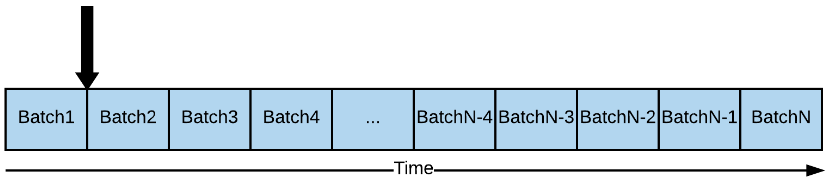


Fig. 1. Illustration of updates and the end of each batch.

to a domain A, the method chooses the model that is trained from items in A. We refer to this method as *context pre-filtering*. Our second approach exploits context for training the model. We refer to this method as *context modeling*. We use the Linear UCB algorithm [10] to exploit context in the model. We use a special case of general contextual bandit algorithm known as *K-armed bandit*, where the set of arms remains the same in all trials. In our framework, the five algorithms introduced earlier in this section are served as the arms. The score of arms is calculated according to Eq.(1) where the feature vectors $x_{t,\alpha}$ are built based on the given context.

For the contextual bandit algorithm, it is important to find useful context features. Unlike other work [8, 10], where the user profile is available, we do not have any information about the user profile. Users can only be tracked by their cookies, and some users even do not allow any tracking. All contextual information is encoded in the incoming messages. In addition, not all context features can contribute to the performance of recommender algorithms. We consider two context features, one is *Domain*, which is also considered as an important context feature in the work [6] and the other is *User-Domain Impressions*. *Domain* refers to different publishers, and *User-Domain Impressions* refers to the number of impressions the user has with the domain.

Figure 2 shows the recommender algorithms' performance with respect to different context features using the CTR metric. In domain '418', a general news portal, the recommender *Click* performs the best. This recommender also shows a good performance in domain '1677', which is also a general news portal. However, the popularity based algorithm does not perform well in domain '35774', a sports news portal. The recommender algorithm *Recent* shows the best performance in this domain. This can be explained by the fact that people always prefer the newest sports news. Also, the bad performance of *PRCate* might be due to the reason the sports domain is already very specific, and further specific category would not help. The findings are consistent with the findings in previous work. Figure 3 shows the algorithms performance with respect to the number of user-domain impressions in domain '418'. Normally, with the increase of the number of user-domain impressions, CTR should also be increased as users with more interactions with the publishers would be more willing to click on the recommendations. However, there are some exceptions. With the increase of *User-Domain Impressions*, CTR observed from the *Click* algorithm experience a sharp decrease. The decrease can be explained by the fact people with enough interactions with the publisher have already seen the popular articles and prefer the newest news. In *context modeling*, the categorical context features are encoded by one-hot coding. The context *Domain* is already a categorical feature by itself. For the context *User-Domain Impressions*, we implement some transformation. Specifically, we classifies the number of *User-Domain Impressions* into three groups: $0 \leq$ number of *User-Domain Impressions* < 5, $5 \leq$ number of *User-Domain Impressions* < 10 and $10 \leq$ number of *User-Domain Impressions* < 15.

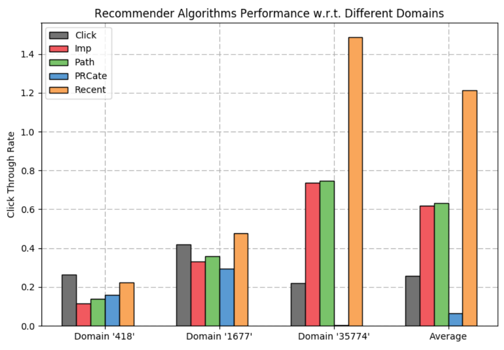


Fig. 2. Recommender Algorithms Performance w.r.t. Different Domains

4 Experiment

In both online and offline scenario, there are four types of messages: *recommendation_request*, *item_update*, *event_notification* and *error_notification*. Specifically, in the Live task, the recommender needs to give recommendations within 100ms, otherwise the recommendations will be regarded as invalid. As the data volume redirected to our recommender is too low in the Live task, the delegation model implemented online is the one with the best performance in NewsREEL Replay. Thus, in the following sections, we will first discuss our experiment in NewsREEL Replay, and then NewsREEL Live.

4.1 NewsREEL Replay

Our recommendation framework that is used in NewsREEL Replay is described in Section 3. For this task, we are provided with a month-long dataset collected

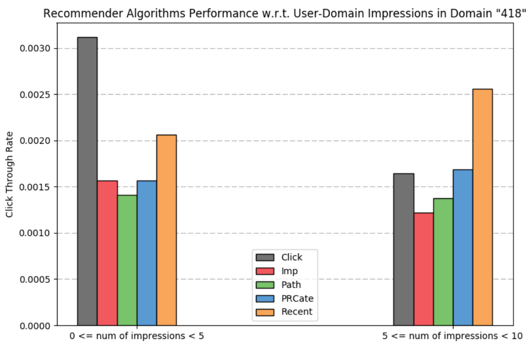


Fig. 3. Recommender Algorithms Performance w.r.t. User-Domain Impressions in Domain "418"

during February 2016. Normally, a replay of a daily dump takes about 3-4 hours. Due to the time constraints, data collected from 2016-02-01 to 2016-02-03 is replayed for evaluation. The dataset is not evenly distributed: about 75% traffic is from the domain '35774', a sports domain, followed by two general news domains '1677' and '418'. In the evaluation phase, we only consider the results from the above three domains since a majority of traffic comes from these three domains. The evaluation phase is as follows: Firstly, the daily dump from 2016-02-01 is replayed to prevent the cold-start problem. Then the evaluation phase is implemented in the following way: The rest of the dataset (data from 2016-02-02 to 2016-02-03) is split into N smaller mutually exclusive datasets with the same size. The dataset is replayed in batches. See Figure 1 for clarification. There are two different batch sizes, one containing 100000 line messages, and the other containing 300000 line messages. Next, as explained in Section 3.1, for each batch, rewards are computed at the end of the batch. After that, the contextual bandit algorithm is able to update the model. Finally, the chosen recommender algorithm is assigned for the incoming requests in the next batch. In Table 1,

Table 1. NewsREEL Replay Evaluation Results: Single Recommender Algorithm Baselines

Algorithm	Average CTR	Domain '418'	Domain '1677'	Domain '35774'
Random	0.645 (0%)	0.187 (0%)	0.401 (0%)	0.749 (0%)
Path	0.632 (−2.02%)	0.137 (−26.74%)	0.357 (−10.97%)	0.748 (−0.13%)
Recent	**1.213 (88.06%)**	0.224 (19.79%)	**0.478 (19.20%)**	**1.487 (98.53%)**
Click	0.256 (−60.31%)	**0.264 (41.18%)**	0.419 (4.49%)	0.220 (−70.63%)
Imp	0.618 (−4.19%)	0.114 (−39.04%)	0.330 (−17.71%)	0.738 (−1.74%)
PRCate	0.064 (−90.08%)	0.159 (−14.97%)	0.294 (−26.68%)	0.03 (−99.60%)

results with the best performance in each domain are highlighted. In Table 2 and 3, results with similar or better performance than the results in Table 1 are also highlighted. For all algorithms, we measure their performance based on average CTR and CTR in the three domains. We also further compare the CTR with the random baseline. The percentage in the tables shows the CTR change comparing to the baseline.

Table 2. NewsREEL Replay Evaluation Results: Context Pre-filtering

Algorithm	Average CTR	CTR '418'	CTR '1677'	CTR '35774'
random	0.645 (0%)	0.187 (0%)	0.401 (0%)	0.749 (0%)
epsilon_0.5_b3	0.954 (47.91%)	0.239 (27.81%)	0.437 (8.98%)	1.148 (53.27%)
epsilon_0.3_b3	1.070 (65.89%)	0.252 (34.75%)	0.468 (16.71%)	1.294 (72.76%)
epsilon_0.1_b3	1.163 (80.31%)	0.235 (25.67%)	0.456 (13.72%)	1.423 (89.99%)
ucb1_b3	1.197 (85.58%)	0.200 (6.95%)	0.420 (4.74%)	1.474 (96.80%)
ts_b3	**1.213 (88.06%)**	0.224 (19.79%)	**0.480 (19.70%)**	**1.486 (98.40%)**
epsilon_0.5_b1	0.958 (48.53%)	0.232 (24.06%)	0.449 (11.97%)	1.151 (53.67%)
epsilon_0.3_b1	1.060 (64.34%)	0.249 (33.16%)	0.458 (14.21%)	1.283 (71.30%)
epsilon_0.1_b1	1.160 (79.84%)	**0.266 (42.25%)**	0.456 (13.72%)	1.415 (88.92%)
ucb1_b1	1.195 (85.27%)	0.199 (6.42%)	0.405 (1.00%)	1.475 (96.93%)

[1] Algorithm labels: *epsilon_0.5_b3:* epsilon_greedy with $\epsilon = 0.5$ with context *Domain* and batch size 300000, *ucb1_b3:* UCB1 with context *Domain* and batch size 300000, *ts_b3:* Thompson Sampling with context *Domain* and batch size 300000

With the context given in Section 3.2, we propose two different delegation models in the Replay task. In the first method, referred as *context pre-filtering*, we only consider the context *Domain* and implement a separate bandit model in each of the three news domains. In the second approach, referred as *context-modeling*, we encode context (*Domain* and *User-domain Impressions*) as feature vectors. Then we build a contextual bandit model based on the linucb (Linear UCB) model. The results of the simple baseline recommender algorithms are presented in Table 1, the results of the context pre-filtering model in Table 2, and the results of the context-modeling approach is listed in Table 3. The random algorithm, which randomly selects a recommender algorithm to serve the incoming requests, is set as the baseline, referred to as *random* in Table 1. The names of the contextual bandit algorithms in Table 2 contain *algorithm_hyperparamter value_batch size.* There are three different bandit algorithms in context pre-filtering: epsilon-greedy (epsilon), ucb1 and Thompson Sampling (ts). For epsilon-greedy, we tried three different ϵ value: 0.5, 0.3 and 0.1. In the experiment, we also tried two batch sizes: b3 means each batch containing 300000 messages and b1 indicating 100000 messages. Contextual bandit algorithms in Table 3 are formatted as *algorithm_hyperparamter value_context used_batch size.* The only algorithm used in context-modeling is linear UCB. In the model, linear UCB is implemented with three different β value in Eq.(1): 1,

Table 3. NewsREEL Replay Evaluation Results: Context-Modeling

Algorithm	Average CTR	CTR '418'	CTR '1677'	CTR '35774'
linucb_1_d_b3	1.202 (86.36%)	0.220 (17.65%)	0.403 (0.50%)	**1.483 (98.00%)**
linucb_0.5_d_b3	1.207 (87.13%)	0.246 (31.55%)	0.423 (5.49%)	**1.483 (98.00%)**
linucb_0.2_d_b3	1.204 (86.67%)	0.257 (37.43%)	0.414 (3.24%)	**1.484 (98.13%)**
linucb_1_di_b3	1.211 (87.75%)	0.259 (38.50%)	0.451 (12.47%)	**1.485 (98.26%)**
linucb_0.5_di_b3	1.206 (86.98%)	**0.279 (49.20%)**	0.415 (3.49%)	**1.485 (98.26%)**
linucb_0.2_di_b3	**1.214 (88.22%)**	**0.268 (43.32%)**	0.467 (16.46%)	**1.485 (98.26%)**
linucb_1_d_b1	0.992 (53.80%)	0.244 (30.48%)	0.452 (12.72%)	1.180 (57.54%)
linucb_0.5_d_b1	1.205 (86.82%)	0.231 (23.53%)	0.457 (13.97%)	1.478 (97.33%)
linucb_0.2_d_b1	1.196 (85.43%)	**0.263 (40.64%)**	0.393 (−2.00%)	1.478 (97.33%)
linucb_1_di_b1	1.196 (85.43%)	0.253 (25.29%)	0.395 (−1.50%)	1.478 (97.33%)
linucb_0.5_di_b1	1.196 (85.43%)	0.207 (10.70%)	0.426 (6.23%)	1.477 (97.20%)
linucb_0.2_di_b1	1.197 (85.58%)	0.150 (−19.79%)	0.456 (13.72%)	1.479 (97.46%)

[1] Algorithm annotation: *linucb_1_d_b3:* LinUCB with $\beta = 1$ with context *Domain* and batch size 300000, *linucb_1_di_b3:* LinUCB with $\beta = 1$ with context *Domain*, *User-Domain Impressions* and batch size 300000

0.5 and 0.2, and two context: *Domain* and *User-Domain Impressions*. In Table 2 and Table 3, *d* stands for the context *Domain*. In Table 3, *di* indicates the use of both contexts.

4.2 NewsREEL Live

The online evaluation of NewsREEL Live is from 24 April to 7 May 2017. The Open Recommendation Platform (ORP) is responsible for redirecting the traffic from the publishers and monitoring different recommender algorithms' performance. With ORP, participants are able to deploy their recommender services and receive recommendation requests. The low volume of data refers to the small number of recommendation requests, user clicks and user impressions, redirected to our recommender from plista. Comparing with the data stream in the Replay dataset, the data stream redirected in the Live task consists only a small portion of the whole plista data stream and is not enough to support training or updating of our contextual bandit algorithm. In the Live task, we deployed the best delegation recommender model trained from the NewsREEL Replay, referring to *linucb_0.2_di_b3* in Table 3, to serve the incoming recommendation requests. In addition, a portion of data from the Live task comes from a domain that is not included in the Replay dataset. This also gives rise to a challenge for us. Our solution is that for all the recommendation requests from that domain, we redirect them to the *PRCate* recommender. Table 4 shows our online evaluation results. There are two types of impressions created by ORP. The first type considers the impressions from the recommendation list, which are referred to as 'widget impressions', and the second type considers the impressions from individual items, which are referred to as 'item impressions'. ORP also creates a

Table 4. NewsREEL Live Evaluation Results from 24 April to 7 May 2017 (delegation model: *linucb_0.2_di_b3* trained from offline)

Metric	Click Num	Impression Num	CTR
Widget	747	60814	0.012
Individual item	747	192207	0.0039

'click' for each click on the recommended articles from users. The final metric for the recommender performance is measured by the CTR, which equals to the number of clicks divided by the number of impressions.

5 Discussion

5.1 NewsREEL Replay

The NewsREEL Replay evaluation results are summarized in Table 1, Table 2 and Table 3. Table 1 shows the recommendation performance of the five basic recommenders from which the delegation model is able to choose an appropriate recommender. A random algorithm, which randomly selects a recommender for the incoming recommendation requests, is served as the baseline. Table 2 shows the evaluation results from the contextual bandit delegation model in which a MAB model is set up for each context. Table 3 shows the evaluation results from the contextual bandit delegation model in which the linear UCB model is implemented. The only context in Table 2 is *Domain*, while in Table 3, two contexts *Domain* and *User-Domain Impressions* are considered. From both Table 2 and Table 3, we can see the evaluation results from the delegation model outperform the random baseline. This indicates the delegation model is able to learn from the past and chooses appropriate recommender for the incoming recommendation requests in order to maximize the CTR.

To further explain the results, we first describe two facts with respect to the domain '35774': Firstly, as explained in the previous section, about 70% of the traffic is from this domain. Therefore, a CTR increase in domain '35774' leads to an increase of average CTR. Secondly, as shown in Table 1, the recommender *Recent* performs much better than any other recommender in this domain. For the performance of different MAB algorithms, UCB1 works better than epsilon-greedy in domain '35774', but worse in domain '418' and domain '1677'. This result is a little bit unexpected since UCB1 explores under the guidance of a confidence bound [10], while epsilon-greedy explores randomly without any guidance. The reason might be the parameter that is used to balance the exploration and exploitation is fixed in UCB1. In the future, we will try UCB with other value of the parameter. Thompson Sampling also works well in the evaluation, and in domain '1677' it shows the best performance. For domain '35774', further exploration does not benefit the CTR since the *Recent* recommender always performs best in the domain. The exploration leads to a CTR decrease in domain '35774' but CTR increases in domain '418' and domain '1677'. For epsilon-greedy, smaller ϵ value shows better results. From the side of epsilon-greedy, exploring too much is also not a good idea. The fixed parameter ϵ is served to balance the exploration and exploitation. In Table 3, in addition to the context *Domain*, another context *User-domain Impressions*, which measures the number of interactions the user has with the domain, is also taken into consideration. The results in Table 3 are in general better than the results from Table 2. The contextual bandit delegation model based on linear UCB works better than the one in which there is a MAB model for each context. In addition, the context *User-domain Impression* is a useful context concerning CTR. Average CTR from the delegation model based on the *linucb_0.2_di_b3* algorithm even outperforms the *Recent* Recommender in Table 1. One interesting observation is that CTR does not increase in domain '35774' and '1677' indicating that the exploration is less useful, but it does increase in domain '418', in which exploration is much more useful. This is due to some recommender algorithms always achieving the best performance in some domains (e.g., *Recent* recommender in domain '35774'), making exploration is unnecessary. The findings also indicate the performance of a single recommender algorithm appears to depend on the publishers.

5.2 NewsREEL Live

For NewsREEL Live we test our recommender for 14 days. There are two challenges in the Live task: Firstly, as discussed before, the low volume of data from the ORP to our account makes it hard for us to train the contextual bandit algorithm online. A delegation model trained from the Replay task is deployed to serve the incoming recommendation requests. Secondly, publishers in the Live task are different from the publishers in the Replay task. In NewsREEL Live, a portion of requests comes from domain '17614', which is not included in the Replay dataset. As the result, the corresponding requests from that domain are delegated to the *PRCate* recommender. The two challenges might lower our recommendation performance in the Live task. In total 19 teams which participate in NewsREEL Live, our delegation model ranked 10th.

5.3 Conclusion and Future Work

In conclusion, our results have verified our assumption the contextual bandit algorithm is able to select the appropriate recommender algorithm given the context. However, the balance between exploration and exploitation is important. If one particular recommender algorithm is performing significantly better, then exploitation does not necessarily help. For future work, firstly, we will add more recommender algorithms to our recommender algorithm pool, such as collaborative filtering since the performance of the contextual bandit algorithm is to some extent limited by the single recommender algorithm. Secondly, we will explore more contexts in order to further improve the recommendation CTR. Last but not least, we would like to evaluate the contextual bandit algorithm on a dataset with a more equally distributed traffic among different domains.

References

1. Hopfgartner, F., Brodt, T., Seiler, J., Kille, B., Lommatzsch, A., Larson, M., Turrin, R., Serény, A.: Benchmarking news recommendations: The CLEF NewsREEL use case. In: ACM SIGIR Forum, ACM (2016) 129–136
2. Lommatzsch, A., Kille, B., Hopfgartner, F., Larson, M., Brodt, T., Seiler, J., Özgöbek, Ö.: CLEF 2017 NewsREEL overview: A stream-based recommender task for evaluation and education. In: Proceedings of CLEF 2017, Dublin, Ireland, 2017. (2017)
3. Kille, B., Lommatzsch, A., Gebremeskel, G.G., Hopfgartner, F., Larson, M., Seiler, J., Malagoli, D., Serény, A., Brodt, T., De Vries, A.P.: Overview of NewsREEL'16: Multi-dimensional evaluation of real-time stream-recommendation algorithms. In: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer (2016) 311–331
4. Hopfgartner, F., Kille, B., Lommatzsch, A., Plumbaum, T., Brodt, T., Heintz, T.: Benchmarking news recommendations in a living lab. In: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer (2014) 250–267
5. Kille, B., Lommatzsch, A., Turrin, R., Serény, A., Larson, M., Brodt, T., Seiler, J., Hopfgartner, F.: Stream-based recommendations: Online and offline evaluation as a service. In: International Conference of the Cross-Language Evaluation Forum for European Languages, Springer (2015) 497–517
6. Lommatzsch, A., Albayrak, S.: Real-time recommendations for user-item streams. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM (2015) 1039–1046
7. White, J.: Bandit algorithms for website optimization. O'Reilly (2012)
8. Tang, L., Rosales, R., Singh, A., Agarwal, D.: Automatic ad format selection via contextual bandits. In: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, ACM (2013) 1587–1594
9. Li, W., Wang, X., Zhang, R., Cui, Y., Mao, J., Jin, R.: Exploitation and exploration in a performance based contextual advertising system. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2010) 27–36
10. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th international conference on World Wide Web, ACM (2010) 661–670
11. Burtini, G., Loeppky, J., Lawrence, R.: A survey of online experiment design with the stochastic multi-armed bandit. arXiv preprint arXiv:1510.00757 (2015)
12. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2-3) (2002) 235–256
13. Chapelle, O., Li, L.: An empirical evaluation of Thompson sampling. In: Advances in neural information processing systems. (2011) 2249–2257
14. Agrawal, S., Goyal, N.: Analysis of Thompson sampling for the multi-armed bandit problem. In: COLT. (2012) 39–1
15. Yuan, J., Lommatzsch, A., Kille, B.: Clicks pattern analysis for online news recommendation systems. In Balog, K., Cappellato, L., Ferro, N., Macdonald, C., eds.: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016. Volume 1609 of CEUR Workshop Proceedings., CEUR-WS.org (2016) 679–690

Bibliography

[1] J. Liu, P. Dolan, and E. R. Pedersen, *Personalized news recommendation based on click behavior,* in *Proceedings of the 15th international conference on Intelligent user interfaces* (ACM, 2010) pp. 31–40.

[2] A. S. Das, M. Datar, A. Garg, and S. Rajaram, *Google news personalization: scalable online collaborative filtering,* in *Proceedings of the 16th international conference on World Wide Web* (ACM, 2007) pp. 271–280.

[3] J. Yuan, A. Lommatzsch, and B. Kille, *Clicks pattern analysis for online news recommendation systems.* in *CLEF (Working Notes)* (2016) pp. 679–690.

[4] P. Beck, M. Blaser, A. Michalke, and A. Lommatzsch, *A system for online news recommendations in real-time with Apache Mahout,* in *Working Notes of the 8th International Conference of the CLEF Initiative, Dublin, Ireland. CEUR Workshop Proceedings* (2017).

[5] P. Probst and A. Lommatzsch, *Optimizing a scalable news recommender system.* in *CLEF (Working Notes)* (2016) pp. 669–678.

[6] A. Lommatzsch, *Real-time news recommendation using context-aware ensembles.* in *ECIR*, Vol. 14 (Springer, 2014) pp. 51–62.

[7] A. Lommatzsch and S. Albayrak, *Real-time recommendations for user-item streams,* in *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (ACM, 2015) pp. 1039–1046.

[8] A. Lommatzsch, N. Johannes, J. Meiners, L. Helmers, and J. Domann, *Recommender ensembles for news articles based on most-popular strategies.* in *CLEF (Working Notes)* (2016) pp. 657–668.

[9] F. Hopfgartner, T. Brodt, J. Seiler, B. Kille, A. Lommatzsch, M. Larson, R. Turrin, and A. Serény, *Benchmarking news recommendations: The CLEF NewsREEL use case,* in *ACM SIGIR Forum* (ACM, 2016) pp. 129–136.

[10] A. Lommatzsch, B. Kille, F. Hopfgartner, M. Larson, T. Brodt, J. Seiler, and Ö. Özgöbek, *CLEF 2017 NewsREEL overview: A stream-based recommender task for evaluation and education,* in *Proceedings of CLEF 2017, Dublin, Ireland, 2017* (2017).

[11] B. Kille, A. Lommatzsch, G. G. Gebremeskel, F. Hopfgartner, M. Larson, J. Seiler, D. Malagoli, A. Serény, T. Brodt, and A. P. De Vries, *Overview of NewsREEL'16: Multi-dimensional evaluation of real-time stream-recommendation algorithms,* in *International Conference of the Cross-Language Evaluation Forum for European Languages* (Springer, 2016) pp. 311–331.

[12] F. Hopfgartner, B. Kille, A. Lommatzsch, T. Plumbaum, T. Brodt, and T. Heintz, *Benchmarking news recommendations in a living lab,* in *International Conference of the Cross-Language Evaluation Forum for European Languages* (Springer, 2014) pp. 250–267.

[13] B. Kille, A. Lommatzsch, R. Turrin, A. Serény, M. Larson, T. Brodt, J. Seiler, and F. Hopfgartner, *Stream-based recommendations: Online and offline evaluation as a service,* in *International Conference of the Cross-Language Evaluation Forum for European Languages* (Springer, 2015) pp. 497–517.

[14] L. Li, W. Chu, J. Langford, and R. E. Schapire, *A contextual-bandit approach to personalized news article recommendation,* in *Proceedings of the 19th international conference on World wide web* (ACM, 2010) pp. 661–670.

[15] C. C. Aggarwal, *Ensemble-based and hybrid recommender systems,* in *Recommender Systems* (Springer, 2016) pp. 199–224.

[16] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, *Recommender systems survey,* Knowledge-based systems **46**, 109 (2013).

[17] G. Adomavicius and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,* IEEE transactions on knowledge and data engineering **17**, 734 (2005).

[18] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, *et al.*, *Collaborative filtering recommender systems,* Foundations and Trends in Human–Computer Interaction **4**, 81 (2011).

[19] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, *A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition,* Information Sciences **180**, 4290 (2010).

[20] G. Adomavicius and A. Tuzhilin, *Context-aware recommender systems,* in *Recommender systems handbook* (Springer, 2011) pp. 217–253.

[21] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, *Fast context-aware recommendations with factorization machines,* in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (ACM, 2011) pp. 635–644.

[22] R. Pagano, P. Cremonesi, M. Larson, B. Hidasi, D. Tikk, A. Karatzoglou, and M. Quadrana, *The contextual turn: from context-aware to context-driven recommender systems.* in *RecSys* (2016) pp. 249–252.

[23] R. Turrin, A. Condorelli, P. Cremonesi, and R. Pagano, *Time-based TV programs prediction,* in *1st Workshop on Recommender Systems for Television and Online Video at ACM RecSys* (2014).

[24] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber, *Offline and online evaluation of news recommender systems at swissinfo.ch,* in *Proceedings of the 8th ACM Conference on Recommender systems* (ACM, 2014) pp. 169–176.

[25] R. Weber *et al.*, *On the gittins index for multiarmed bandits,* The Annals of Applied Probability **2**, 1024 (1992).

[26] A. Zheng, *Evaluating Machine Learning Models* (O'Reilly, 2015).

[27] J. White, *Bandit algorithms for website optimization* (O'Reilly, 2012).

[28] L. Tang, R. Rosales, A. Singh, and D. Agarwal, *Automatic ad format selection via contextual bandits,* in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (ACM, 2013) pp. 1587–1594.

[29] W. Li, X. Wang, R. Zhang, Y. Cui, J. Mao, and R. Jin, *Exploitation and exploration in a performance based contextual advertising system,* in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2010) pp. 27–36.

[30] 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).

[31] P. Auer, N. Cesa-Bianchi, and P. Fischer, *Finite-time analysis of the multiarmed bandit problem,* Machine learning **47**, 235 (2002).

[32] O. Chapelle and L. Li, *An empirical evaluation of Thompson Sampling,* in *Advances in neural information processing systems* (2011) pp. 2249–2257.

[33] S. Agrawal and N. Goyal, *Analysis of Thompson Sampling for the multi-armed bandit problem,* in *COLT* (2012) pp. 39–1.

[34] D. Agarwal, *Recommending items to users: An explore/exploit perspective.* in *UEO@ CIKM* (2013) pp. 1–2.

[35] H. Zheng, D. Wang, Q. Zhang, H. Li, and T. Yang, *Do clicks measure recommendation relevancy?: an empirical user study,* in *Proceedings of the fourth ACM conference on Recommender systems* (ACM, 2010) pp. 249–252.

[36] A. Said, D. Tikk, K. Stumpf, Y. Shi, M. Larson, and P. Cremonesi, *Recommender systems evaluation: A 3D benchmark.* in *RUE@ RecSys* (2012) pp. 21–23.

[37] *Plista client,* https://github.com/recommenders/plistaclient, accessed 9 January 2017.

[38] *Plista recommender,* https://github.com/recommenders/plistarecs, accessed 9 January 2017.

[39] *ORP Protocol,* https://orp.plista.com/documentation/download, accessed 15 February 2017.

[40] *Code to the book "bandit algorithms for website optimization",* https://github.com/johnmyleswhite/BanditsBook, accessed 10 June 2017.

[41] *Contextual bandit Python library,* https://github.com/ntucllab/striatum, accessed 23 March 2017.

[42] *CLEF NewsREEL News Recommendation Evaluation Lab, tasks,* http://www.clef-newsreel.org/tasks/, accessed 14 June 2017.

[43] B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz, *The plista dataset,* in *Proceedings of the 2013 International News Recommender Systems Workshop and Challenge* (ACM, 2013) pp. 16–23.

[44] P. Bons, N. Evans, P. Kampstra, and T. van Kessel, *A news recommender engine with a killer sequence,* in [47].

[45] G. G. Gebremeskel and A. P. de Vries, *Recommender systems evaluations: Offline, online, time and A/A test.* in *CLEF (Working Notes)* (2016) pp. 642–656.

[46] Y. Liang, B. Loni, and M. Larson, *CLEF NewsREEl 2017: Contextual bandit news recommendation,* in [47].

[47] L. Cappellato, N. Ferro, L. Goeuriot, and T. Mandl, eds., *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017*, CEUR Workshop Proceedings, Vol. 1866 (CEUR-WS.org, 2017).