Improved User Cold-Start Recommendation via Two-Level Bandit Algorithms

Otavio Augusto Rodrigues[1], **Anisio Lacerda**[1], **Flávio Luis Cardeal Pádua**[1]

[1]Departamento de Computação
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Av. Amazonas, 7675 – Nova Gameleira – Belo Horizonte – MG – Brazil

otavio.augusto@outlook.com, {anisio,cardeal}@decom.cefetmg.br

Abstract. *Recommender systems have been gaining visibility over the years due to its ability in helping users to deal with the information overload problem. However, the recommendation to new users (user cold-start) is still an open problem for these systems. The user cold-start problem is recognized as an exploration/exploitation dilemma, for which the system needs to balance (i) maximizing the user satisfaction (exploitation) and (ii) learn about users' tastes (exploration). We propose a new bandit algorithm that suggests items to users as a two-step process. First, we select the most relevant cluster to the target user, then, given the selected cluster, we choose an item from it that matches the user's tastes. The experimental evaluation shows that our strategy yields significant improvements regarding recommendation quality over the state-of-the-art bandit algorithms.*

1. Introduction

The aiming of recommender systems is to recognize relevant items according to user's tastes. Recommender systems are based on users' explicit and implicit tastes, the preferences of other users, and user and item attributes. For instance, a movie recommender system can incorporate both explicit rating data (e.g., Alice rates *Toy Story* a 4 out of 5), and movie content information (e.g., *Forrest Gump* is targeted as comedy) to suggest movies that match users' tastes.

There is inherent uncertainty in the user's preference information, which can be gathered through online interaction with users and real-time adaptation of recommendation models. Furthermore, a considerable number of users might be completely new to the recommender system, i.e., there is insufficient information about consumption history. This scenario is known as the user *cold-start* problem [Schein et al. 2002] and represents a challenging problem for recommender systems. In a user *cold-start* setting, traditional recommendation approaches suffers from learning a matching function for users' tastes and items popularity.

In Figure 1, we present the distribution of ratings per users for the MovieLens1M dataset [Miller et al. 2003]. As we can see, few users rates (i.e., evaluates) a large number of movies; whereas, a lot of users rates a small number of movies. This distribution is an example of the user cold-start problem, for which there is severe scarcity in the preferences from users to items.

The user cold-start scenario represents a challenge to recommender systems, and are recognized as an exploration/exploitation problem. This problem refers to find a trade-

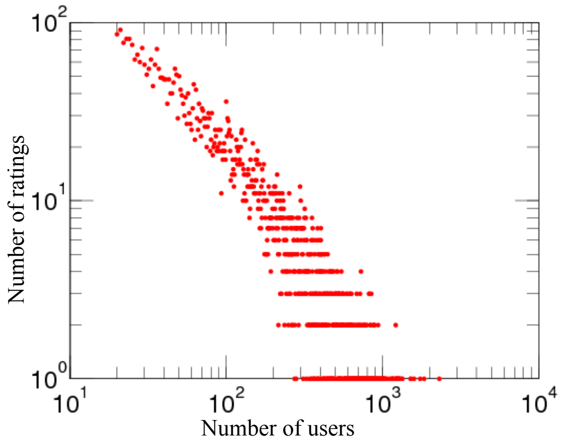

Figure 1. Distribution of ratings per users.

off between two competing goals: (i) maximizing users satisfaction, in the long run, while (ii) exploring uncertainties of user tastes [Agarwal et al. 2009]. For instance, a movie recommender system should find the most relevant movie to users while keep trying to improve the understanding of their preferences.

Traditionally, the exploration/exploitation dilemma is formulated as a multi-armed problem [Li et al. 2010, Tang et al. 2013, Vermorel and Mohri 2005]. In our context, i.e., recommender systems, we state the problem as: for each recommendation round, the algorithm selects an item (e.g., a movie) to pull (i.e., recommend), and after pulling, it receives a reward. Hence, personalized recommendation can be seen as an instantiation of the multi-armed bandit problem. Note that the reward is drawn from an unknown probability distribution determined by the selected item, and it refers to the user response (e.g., a click). The reward is fed back to the bandit algorithm and used to optimize its recommendation strategy. Shortly, the optimal strategy is to pull the arm with the maximum expected reward with respect on the user on each round, and thus maximize the total cumulative reward for the set of rounds.

Typical solutions of the multi-armed bandit problem assume that the arms are independent [Auer et al. 2002]. In this work, we drop this assumption and focus on the bandit problem for which the arms are *dependent*. This dependent-arm assumption has practical scenarios of interest. For instance, we can cast the news recommendation problem as a bandit problem where each news article refers to an arm, showing an article corresponds to pulling an arm, and user clicks are the reward. News articles with similar textual content are likely to receive similar amount of clicks, and this creates dependencies between the arms.

In this paper, to capture the dependence among items, we propose a generalized two-level clustering-based bandit framework. More specifically, first, we consider the arms as clusters of items and use a bandit algorithm to select a group of items to be recommended. Hence, given a pulled cluster, we use a distinct bandit algorithm to select the most relevant item within this cluster. Our proposed framework is flexible to consider, as input, (i) a clustering method and (ii) two bandit algorithms (i.e., one for each level), which leads to several algorithms as instantiations of the framework.

In summary, the contribution of this work is twofold:

- We propose a new bandit algorithm to deal with user cold-start, which is based on a two-level recommendation decision process of clustered items.
- We conduct extensive experiments on a real-world movie dataset to validate the accuracy of the proposed family of algorithms.

The paper is organized as follows. In Section 2, we present the related previous work. In Section 3, we detail our proposed bandit algorithm for the recommendation in user cold-start scenario. In Section 4, we present the experimental setup. In Section 5, we list and discuss the achieved results. Finally, in Section 6, we conclude the paper and present possible future lines of research.

2. Related work

In the following, we highlight the previous research that is most relevant to our work.

Bandits for recommendations: In the recommender systems literature, there is an extensive literature of methods conceived to perform online recommendations. As already mentioned, the most common approach to deal with online recommendation is modeling the task as a multi-armed bandit problem [Bouneffouf et al. 2012, Li et al. 2010]. In [Li et al. 2010], the authors first introduced the online new recommendation problem using a bandit formulation, which first mention the contextual multi-armed bandit algorithms. In this case, the context refers to information about users. We emphasize that our approach is *context-free*, i.e., we assume that all users are new, for which there is no historic information. Hence, we are unable to compare our approach to contextual multi-armed bandit algorithms, since the later assume that there is historic information about users.

User cold-start problem: The user cold-start problem is one of the most challenge problems in recommender systems. Hence, there have been several approaches proposed to deal with this scenario of scarcity of information [Park et al. 2006, Leroy et al. 2010]. For instance, in [Park et al. 2006], the authors proposed the usage of filterbots, which are bots that automatically inject ratings into the recommender system to reduce data sparsity. Hence, this simple approach inserts "fake" ratings into the system to improve the quality of cold-start recommendations. In a different scenario, i.e. friend suggestion in social networks, the authors in [Leroy et al. 2010], investigated the cold-start problem using group membership preferences among peers. In this work, different from previous approaches, we focus on online recommendation.

3. TL-Bandits

In this section, we formalize the problem investigated in this paper and present relevant concepts critical to understanding the on-line recommendation task.

3.1. Problem Formalization

Formally, let U be an arbitrary input space and $I = \{1, \dots, M\}$ be a set of M arms, where each arm corresponds to an item. Let E be a distribution over tuples (u, r), where $u \in U$ corresponds to the target user, i.e., the user we are interested in recommending items to,

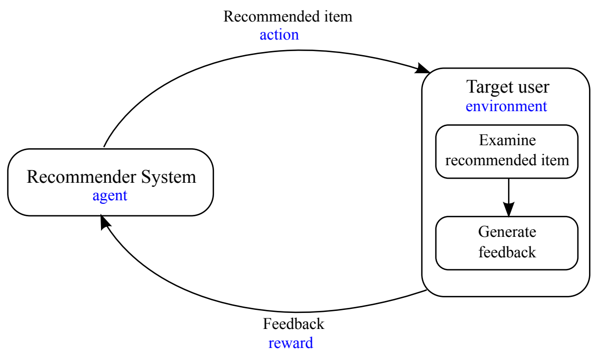

Figure 2. The online recommendation task modeled as a Multi-armed Bandit problem.

and $r \in \{0, 1\}$ refers to the reward received by the algorithm. Hence, the reward is 1 if the user considers that the suggested item is relevant, and 0 otherwise.

In Figure 2, we present the interaction cycle between a target user and an on-line recommender system. The user arrives, and the recommender system selects a relevant item, which is presented to the user. The user interacts with the list by clicking on the suggested item whether it is relevant to him. The interaction is interpreted as feedback about the quality of the recommended item. Hence, this feedback is used to update the recommendation model aiming to provide better recommendations in the future. The cycle finishes and the next user is considered. This formulation translates to the Reinforcement Learning problem (see the terminology in blue color in Figure 2.

3.2. Multi-armed Bandits

The online recommendation task is often modeled as a Multi-armed Bandit problem, which originates from the reinforcement learning paradigm. The MAB problem is stated as a sequential Markov Decision Process (MDP) of an agent that tries to optimize its actions while improving its knowledge on the arms. The central challenge in bandit problems is the need for balancing exploration and exploitation. Interactively, the algorithm $\mathcal{A}$ *exploits* past knowledge to choose the arm that seems best. On contrary, this apparently optimal arm may be suboptimal choice, due to imprecisions in $\mathcal{A}$'s knowledge. To avoid this undesired situation, $\mathcal{A}$ has to *explore* by selecting seemingly suboptimal arms to gather more information about them.

In Algorithm 1, we present the generalized MAB algorithm. A MAB algorithm iterates in discrete rounds $t = 1, 2, \dots$ (Line 1). In each round t, the algorithm selects an item $i(t) \in I$ based on target user $u(t)$ and on the knowledge, it collected from the previous rounds (Line 2) and presents the selected item to the target user (Line 3). Hence, the reward $r(t)$[1] is revealed (Line 4–8). Finally, the algorithm improves its arm-selection policy with the new observation $u_t, i(t), r(t)$ (Line 9). Note that the algorithm chooses a single item per round.

3.3. Two-Level Bandit Algorithms (TL-Bandits)

In Figure 3, we present the overall scheme of our proposed Two-Level Bandit algorithm. As we can see, the selection of which item suggest to the target user is performed in two

[1]We use $r(t)$ as the reward for item $i(t)$ in trial t

Algorithm 1 Generalized Multi-armed Bandits

```
Require: Bandit algorithm $\mathcal{A}$
 1: for $t = 1, \ldots, T$ do
 2:     $i(t) \leftarrow$ select-item$(u(t), \mathcal{A})$
 3:     show $i(t)$ to user; register click
 4:     if user clicked $i(t)$ then
 5:         $r(t) = 1$
 6:     else
 7:         $r(t) = 0$
 8:     end if
 9:     update-mab$(u(t), i(t), r(t), \mathcal{A})$
10: end for
```

steps. First, we need to select the cluster that is most likely to contain a relevant video to the user. Hence, given the selected cluster, we have another step that is the selection of a video within the chosen cluster. In both levels, we have a MAB algorithm that is responsible for selecting a cluster (level 1) or a video (level 2). However, the reward is dependent on the choice made by the MABs.

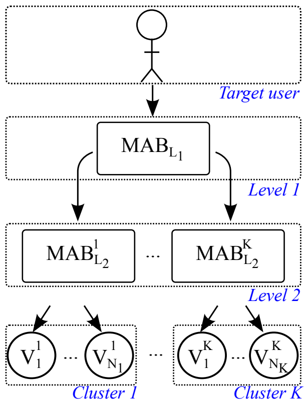

Figure 3. TL-Bandits overview. We assume K clusters of items.

In Algorithm 2, we detail the online recommendation problem as a Two-Level bandit algorithm. Note that our algorithm assumes as input two bandit algorithms, one for each level. We also consider that the items (i.e., videos) were previously clustered. Hence, as in Algorithm 1, the recommendation process iterates in discrete rounds $t = 1, 2, \ldots$ (Line 1). In each recommendation round t, the algorithm first selects a cluster from which an item will be selected (Line 2). In Line 3, the MAB from level 2 then selects a video within the previously selected cluster. The selected video is presented to the target user, and their feedback is captured (Lines 4–9). The final step refers to updating the knowledge for the MABs in each level independently (Lines 10–11). In this work, we assume that the MABs in both levels are from the same family, i.e., if we have the Bayes UCB algorithm in level 1, then we have another instantiation of the same algorithm in level 2. We emphasize that the proposed TL-Bandits are flexible to consider any MAB algorithm to choose both the cluster and the videos within the clusters.

Algorithm 2 Two-Level Bandit Algorithm

```
Require: Bandit algorithm level-1 $\mathcal{A}_1$ and level-2 $\mathcal{A}_2$
 1: for $t = 1, \dots, T$ do
 2:     $c(t) \leftarrow$ select-cluster$(u(t), \mathcal{A}_1)$
 3:     $i(t) \leftarrow$ select-item$(u(t), c(t), \mathcal{A}_2)$
 4:     show $i(t)$ to user; register click
 5:     if user clicked $i(t)$ then
 6:         $r(t) = 1$
 7:     else
 8:         $r(t) = 0$
 9:     end if
10:     update-mab$(u(t), i(t), r(t), \mathcal{A}_1)$
11:     update-mab$(u(t), i(t), r(t), \mathcal{A}_2)$
12: end for
```

4. Experimental setup

In this section, we describe our experimental setup.

4.1. Research Questions

The research questions guiding the remainder of the paper are:

RQ1 Do clustering approaches improve the state-of-the-art of online recommendation methods? (See Section 5.1)

RQ2 What is the effect on recommendation accuracy of varying the clustering method? (See Section 5.2)

RQ3 What is the effect on recommendation accuracy of varying the number of clusterings? (See Section 5.3)

RQ4 What is the effect on recommendation accuracy over the rounds? (See Section 5.4)

4.2. Dataset

To assess the recommendation quality of our method, we used the MovieLens1M dataset, which contains 1,000,209 ratings of 3,883 movies made by 6,040 users [Miller et al. 2003]. The collected ratings are in a 1-to-5 star scale. We emphasize that this is a publicly available movie rating dataset.

4.3. Metrics

In the context of item recommendation, when an item is clicked, a reward of 1 is incurred; otherwise the reward is 0. Hence, the expected reward of an item is its *Click-Through Rate* (CTR), and choosing an item with maximum predicted CTR is the same as to maximizing the expected number of clicks from users, which in turn equivalent to maximizing the total expected reward of our bandit formulation.

The averaged reward is equivalent to the metric CTR, which is the total reward divided by the total number of trials:

$$Relative\ \ CTR = \frac{1}{T}\sum_{t=1}^{T} r_t, \tag{1}$$

where r_t is the reward at round t, and T is the total number of rounds. We report the algorithm *relative CTR*, which is the algorithm's CTR divided by the random recommender.

We also report results based on the *Cumulative Click-Through Rate*, which refers to the sum of rewards received by a MAB algorithm until round i. Formally:

$$Cumulative\ \ CTR(i) = \sum_{t=1}^{i} r_t, \tag{2}$$

In other words, the Cumulative CTR measures the total amount of reward that an algorithm receives up until some fixed point in time.

4.4. Multi-armed Bandit Algorithms

Following, we detail the context-free MAB algorithms used to instantiate our framework.

- Random: it randomly chooses an item to pull.
- ϵ-greedy (ϵ) (EG): it chooses a random item with probability ϵ, and chooses the arm with the highest CTR estimate with probability $(1 - \epsilon)$ [Tokic 2010].
- UCB1: it chooses the item with the highest UCB (*Upper Confidence Value*) score, according to [Auer et al. 2002].
- UCB2 (η): the same as the above, but with a different UCB score computation [Auer et al. 2002]. The η parameter balances the amount of exploration and exploitation.
- Thompson sampling (TS): it is a bayesian implementation of the MAB algorithm in which the (unknown) reward values are inferred from past data and summarized using a posterior distribution. It selects the item proportionally to its probability of being optimal under this posterior [Thompson 1933].
- Bayes UCB: it is a Bayesian adaptation of the UCB algorithm, in which the upper confidence bound score is computed by taking into account an upper quantile on the posterior mean [Kaufmann et al. 2012].

4.5. Clustering Methods

To instantiate our framework, we use three distinct clustering strategies, which are detailed in the following.

4.5.1. Category

In this clustering approach, we use the pre-defined category in which all movies in the dataset are classified. For instance, we have the categories *Action*, *Adventure*, *Romance*, etc. Please, refer to [Miller et al. 2003] to see the complete list of categories. The main disadvantages of this clustering approach are that (i) the number of clusters is fixed, and (ii) it is costly, since we need a manual classification for each item.

4.5.2. k-Means

Let $M = \{m_1, \ldots, m_n\}$ be the set of movie title and synopses concatenated for each movie m_i, for which we want to compute a set of k clusters, $C = c_1, \ldots, c_k$ [Jain 2010].

The k-Means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points within the cluster is minimized. Given μ_k be the centroid of cluster c_k, the squared error $E(c_k)$ between μ_k and the points in cluster c_k is defined as

$$E(c_k) = \sum_{m_i \in c_k} ||m_i - \mu_k||^2. \tag{3}$$

The final goal of k-Means is to minimize the sum of the squared error over all K clusters,

$$E(C) = \sum_{k=1}^{K} \sum_{m_i \in c_k} ||m_i - \mu_k||^2. \tag{4}$$

4.5.3. Latent Dirichlet Allocation (LDA)

The Latent Dirichlet Allocation topic model is a generative probabilistic model for collections of discrete data (e.g., text corpora) [Blei et al. 2003]. A topic model is an algorithm that focuses on discovering latent structures, i.e., hidden patterns that help to describe the data. Since it is a generative model, it defines how words in a document are generated through the control of latent topics. In the context of movie recommendation, the words come from movie title and synopsis, and the topics refer to groups of movies with similar characteristics.

In LDA, it is assumed that each observed movie description (title and synopsis) was generated by weighted mixtures of unobserved latent topics, which can be learned from the documents and refers to semantic themes present in the movie dataset. Formally, the LDA is defined using the following terms [Blei et al. 2003]:

- A *word* is the basic unit of discrete data from a vocabulary V.
- A *movie description* is a sequence of N words extracted from movie's title and synopsis.
- A *dataset* is a collection of $|I|$ movie descriptions.

The generative process of LDA specifies a simple probabilistic procedure to generate new documents according to a set of topics ϕ. The movie description are generated by first selecting a distribution over topics θ from a Dirichlet distribution, which determines the probability $P(z)$ (probability of topic z) for words in that description. The words in the movie description are then generated by selecting a topic i from this distribution and then picking a word from that topic following $P(w|z = i)$. The estimation of model parameters used here follows the Collapsed Gibbs Sampling approach [Griffiths and Steyvers 2004].

4.6. Evaluation Protocol

The evaluation of online learning algorithms with past data is a challenging task. Here, we the state-of-the-art method for offline evaluation of multi-armed bandit algorithms, namely, *BRED* (*Bootstrapped Replay on Expanded Data*) method. The BRED method is based on bootstrapping techniques and enables an unbiased evaluation by utilizing historical data. The method assumes that the dataset is static to perform the bootstrap resamples.

Table 1. Average Relative CTR. In bold we highlight the gains over the baseline. In parentheses we show the number of clusters.

	Baseline	Category	k-Means Clustering						LDA Clustering					
	-	(17)	(5)	(10)	(17)	(25)	(50)	(100)	(5)	(10)	(17)	(25)	(50)	(100)
Random	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
EG(0,0)	1,146	**3,752**	**1,326**	0,233	**2,974**	0,323	0,364	0,010	**2,126**	0,045	0,045	0,138	0,000	0,087
EG(0,1)	1,034	**6,296**	**1,670**	**1,757**	**3,466**	**4,836**	**4,248**	**3,182**	**6,551**	**5,568**	**6,348**	**5,601**	**5,026**	**6,317**
EG(0,2)	0,921	**5,872**	**1,545**	**1,929**	**3,345**	**4,079**	**3,467**	**4,924**	**5,669**	**5,288**	**6,536**	**4,978**	**5,517**	**6,260**
EG(0,3)	0,966	**4,576**	**1,602**	**1,981**	**2,569**	**3,360**	**3,486**	**4,667**	**4,795**	**3,909**	**4,446**	**4,290**	**4,810**	**5,308**
EG(0,4)	0,989	**3,608**	**1,511**	**1,871**	**2,586**	**3,503**	**3,168**	**2,444**	**3,819**	**3,432**	**4,259**	**3,043**	**3,698**	**3,817**
EG(0,5)	0,933	**3,248**	**1,371**	**1,743**	**1,629**	**3,053**	**2,364**	**3,298**	**3,472**	**2,750**	**2,205**	**2,036**	**2,879**	**3,490**
EG(1,0)	0,944	**1,008**	**0,947**	0,929	**1,009**	**1,116**	0,841	0,803	**0,992**	**0,977**	**1,036**	**1,014**	0,931	0,904
UCB1	0,876	**1,824**	0,852	0,833	**1,560**	**0,910**	0,864	**0,955**	**2,835**	**1,818**	**1,411**	**1,275**	**1,328**	**1,452**
UCB2 (0,0)	2,135	**3,752**	1,326	0,233	**2,974**	0,323	0,364	0,010	2,126	0,045	0,045	0,138	1,681	0,087
UCB2 (0,1)	1,011	**2,104**	**1,061**	**1,033**	**1,690**	**1,048**	**1,028**	**1,020**	**3,630**	**2,220**	**1,679**	**1,522**	**1,422**	**1,798**
UCB2 (0,2)	0,831	**2,040**	**1,000**	**1,057**	**1,707**	**1,063**	**1,061**	**1,081**	**3,748**	**2,083**	**1,634**	**1,449**	**1,543**	**1,538**
UCB2 (0,3)	0,978	**2,024**	**1,027**	**1,090**	**1,595**	**1,111**	**1,028**	**0,980**	**3,567**	**2,167**	**1,563**	**1,551**	**1,491**	**1,462**
UCB2 (0,4)	0,933	**2,216**	**1,030**	**0,976**	**1,750**	**1,201**	**0,972**	**0,975**	**3,567**	**2,000**	**1,598**	**1,725**	**1,543**	**1,615**
UCB2 (0,5)	1,135	**2,112**	1,057	0,981	**1,931**	1,106	0,986	1,015	**3,709**	**1,977**	**1,813**	**1,406**	**1,474**	**1,846**
UCB2 (0,7)	1,079	**2,216**	0,932	**1,124**	**1,948**	**1,196**	1,056	1,020	**3,614**	**1,932**	**1,598**	**1,565**	**1,672**	**1,769**
UCB2 (1,0)	1,045	**2,192**	**1,102**	**1,167**	**1,983**	**1,317**	**1,276**	**1,172**	**3,386**	**2,106**	**1,688**	**1,609**	**1,741**	**1,904**
Bayes UCB	0,697	**6,344**	**1,652**	**1,862**	**2,897**	**3,804**	**2,430**	**2,157**	**7,551**	**6,811**	**7,107**	**5,181**	**4,267**	**2,846**
TS	1,079	**1,272**	**1,856**	**2,205**	**1,207**	**4,529**	**2,907**	**3,616**	1,016	0,970	**1,098**	**1,246**	**1,181**	**1,231**

We follow the same approach presented in [Mary et al. 2014], and assume the static world on small portions of the MovieLens1M dataset. We took the smallest number of portions such that a given portion has a fixed number of movies. The BRED evaluation method stems from the idea of bootstrapping, first introduced by [Efron 1979]. Let us consider each portion as a single dataset D of size T with K possible choices (arms). Then, from D, at each step, generate P new sampled datasets $D_1, D_2, ..., D_P$ of size $K \times T$ by sampling with replacement. For each sampled dataset i of size D_i with K_i items, we compute the estimated click through-rate (CTR) by averaging the CTR of each bandit algorithm on 100 random permutations of the data. We report the average CTR and cumulative CTR over all portions.

5. Results and Discussion

5.1. Performance of baseline recommendation methods

In Table 1, we present the CTR performance metric of all tested bandit algorithms. Note that, all results are normalized about the Random method. Hence, strategies with numbers below 1.0 are worst, regarding CTR, than the Random method. The Baseline column refers to the bandit strategies without considering clusters of items, i.e., they refer to strategies for which the arm is the item. In bold we highlight the results for which we achieve a better result than the baselines.

We first consider the clusterings as the category of the movies (e.g., Drama, Comedy, ...). When using the category to clusters the movies, we have a pre-defined number of groups. In our dataset, the movies are grouped into 17 categories. When using this instantiation of our framework, we achieve promising results regarding CTR. For instance, the Bayes UCB algorithm achieves an average relative CTR of $6,344$.

We also tested two other clustering approaches for which we can define the number of clusters, i.e., k-Means and LDA. When analyzing the best results, we can see that

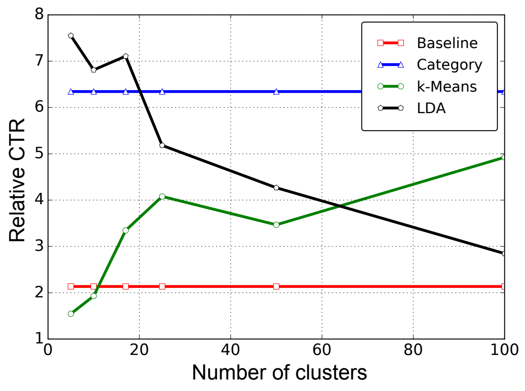

Figure 4. CTR vs Number of Clusters. The MAB algorithm used with Baseline, Category, k-Means, and LDA, are UCB2(0.0), Bayes UCB, EG(0.2), and Bayes UCB, respectively.

the LDA with 5 clusters outperforms all tested configurations of our proposed framework and all instances of the baseline.

5.2. Effect of the clustering methods

As presented in Table 1, the best results regarding CTR refers to clustering items with LDA topic representation. Remembering that LDA extracts topics from movie synopses and represent them in a latent space. We use this representation to describe the movies and cluster them according to the most probable topic. An advantage of LDA and k-Means is that the number of clusters is flexible and can be tuned according to each dataset. We attribute the best results of LDA to its robustness in dealing with the highly sparse description of movies' synopses.

We can also see from Table 1 that the performance of k-Means is better than the baseline. However, when compared with both Category and LDA clustering strategies, the k-Means presents was unable to present better recommendations. We believe that the sparsity of textual representation of items leads to the poor items grouping generated by the k-Means algorithm.

5.3. Effect of the number of clusters

In Figure 4, we present the relative CTR values regarding the number of clusters. Note that both the baselines and the category-based clustering approach are independent of the number clusters. In the former, i.e. the baselines, each item is an arm, hence there are no clusters. In the latter, i.e. the category-based approach, the number of clusters is pre-defined and equal to 17 in our dataset. Hence, for these cluster insensitive approaches, in Figure 4, we have two lines parallel to the x-axis.

When considering the k-means and LDA clustering approaches, we have different patterns. Whereas in k-Means, as we increase the number of clusters, we see improvements in performance; in the LDA, we have the opposite trend, i.e., the best performance of LDA happens when we have only 5 clusters.

5.4. Analysis of recommendation quality through rounds

In this section, we investigate the behavior of each recommendation strategy as the number of rounds evolves. From Figure 5, note that each round refers to a specific interaction

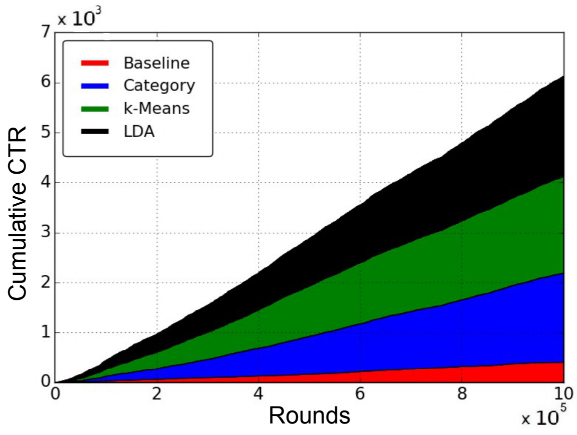

Figure 5. Cumulative Reward. The MAB algorithm used with Baseline, Category, k-Means, and LDA, are UCB2(0.0), Bayes UCB, EG(0.2), and Bayes UCB, respectively.

between the target user and the recommender system. The LDA clustering approach combined with the Bayes UCB MAB algorithm reaches the best performance through the rounds.

5.5. Reproductibility

The dataset we have used in our experiments is publicly available and can be obtained in [Miller et al. 2003]. All base algorithms used here and our implementations are also freely available. Thus, we emphasize that our results are easily reproducible, which enables future developments and possible improvements over our proposed framework.

6. Conclusion and future work

In this work, we propose a new multi-armed bandit algorithm to deal with the user cold-start problem, which works in two steps. The TL-BANDITS framework is flexible and may be instantiated in different ways depending on the scenario of application. We conducted several experiments using a real-world dataset of movie recommendation. The experimental results show that our framework can leverage the state-of-the-art context-free multi-armed bandit algorithms and yield to significant gains over them. As future work, we intend to investigate other combinations of bandit algorithms for each level of the TL-BANDITS framework. We also believe that an appealing line of research is using characteristics of the users to choose the best cluster.

Acknowledgements

The authors gratefully acknowledge the financial support of CEFETMG; CNPq under Proc. 431458/2016-2; and FAPEMIG under Procs. Universal FAPEMIG, process number APQ-03445-16, and FAPEMIG-PRONEX-MASWeb, Models, Algorithms and Systems for the Web, process number APQ-01400-14.

References

[Agarwal et al. 2009] Agarwal, D., Chen, B.-C., Elango, P., Motgi, N., Park, S.-T., Ramakrishnan, R., Roy, S., and Zachariah, J. (2009). Online models for content optimization. In *Advances in Neural Information Processing Systems*, pages 17–24.

[Auer et al. 2002] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

[Blei et al. 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022.

[Bouneffouf et al. 2012] Bouneffouf, D., Bouzeghoub, A., and Gançarski, A. (2012). A contextual-bandit algorithm for mobile context-aware recommender system. In *NIPS*, pages 324–331.

[Efron 1979] Efron, B. (1979). Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26.

[Griffiths and Steyvers 2004] Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *National Academy of Sciences*, 101(suppl 1):5228–5235.

[Jain 2010] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666.

[Kaufmann et al. 2012] Kaufmann, E., Cappé, O., and Garivier, A. (2012). On bayesian upper confidence bounds for bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pages 592–600.

[Leroy et al. 2010] Leroy, V., Cambazoglu, B. B., and Bonchi, F. (2010). Cold start link prediction. In *ACM SIGKDD*, pages 393–402.

[Li et al. 2010] Li, L., Chu, W., Langford, J., and Schapire, R. (2010). A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670.

[Mary et al. 2014] Mary, J., Preux, P., and Nicol, O. (2014). Improving offline evaluation of contextual bandit algorithms via bootstrapping techniques. In *ICML*, pages 172–180.

[Miller et al. 2003] Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., and Riedl, J. (2003). Movielens unplugged: experiences with an occasionally connected recommender system. In *ICIUI*, pages 263–266.

[Park et al. 2006] Park, S.-T., Pennock, D., Madani, O., Good, N., and DeCoste, D. (2006). Naïve filterbots for robust cold-start recommendations. In *ACM SIGKDD*, pages 699–705.

[Schein et al. 2002] Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *ACM SIGIR*, pages 253–260.

[Tang et al. 2013] Tang, L., Rosales, R., Singh, A., and Agarwal, D. (2013). Automatic ad format selection via contextual bandits. In *CIKM*, pages 1587–1594.

[Thompson 1933] Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294.

[Tokic 2010] Tokic, M. (2010). Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *KI 2010: Advances in Artificial Intelligence*, pages 203–210. Springer.

[Vermorel and Mohri 2005] Vermorel, J. and Mohri, M. (2005). Multi-armed bandit algorithms and empirical evaluation. In *ECML*, pages 437–448.