Recommender system with serendipity

Clémence Réda
École Normale Supérieure Paris-Saclay
supervised by Claire Vernade

Abstract

A lacking feature of current recommender systems is that they usually do not allow discovery of new elements, that is, an element that might be interesting, but which is different of what the user is *used to*. This feature is subject to active research in the domain since 2001, because diversity of recommendation (in a general sense) might be useful in order to avoid overfitting or bias, or boredom of the user. Although the subject of diversity/discovery/... in recommender systems is not new, to the best of my knowledge, there is no formal, satisfactory definition of *serendipity* (that is, a surprisingly good discovery) in the context of recommender systems. The goal of this project was (1) to formalize the problem of recommendation allowing serendipities; (2) to design a method which solves this problem in an online setting; (3) to evaluate its relevance with respect to naive and classic bandit models.

1 Introduction

Recommender systems have a wide range of applications, from e-commerce (Smith and Linden [2017]) to educational data mining (Tang and McCalla [2003]). This may account for the high research activity in this field since the early 1980s, and many sorts of methods were developed (see Adomavicius and Tuzhilin [2005] for a comprehensive survey of early algorithms). Many researchers (Abbassi et al. [2009], Kunaver and Požrl [2017], Zhou et al. [2010])have already stressed the difference between *accurate* recommendations (that is, the user actually enjoys the recommended item, and buys it or rates it positively), and *useful* recommendations (recommended items which are novelties, discoveries to the user things that they would never had found themselves). If a recommender system for e-commerce suggests popular items, for instance, most of the users might buy them, but, on the one hand, more obscure books, for instance, might be ignored, thus might never be sold, even if there are users who would be interested in buying them; on the other hand, there might be a drop in sales once the fad for a given item is over, or when most of the customers had already acquired it. The need for diversification of recommender systems is thus strong, and has been a subject of research since 2001 (Bradley and Smyth [2001]) (see Kunaver and Požrl [2017] for a recent survey of this sub-field).

This "diversity-accuracy dilemma of recommender systems" (Zhou et al. [2010]) bears more than a passing resemblance to the exploration-exploitation dilemma that is notably observed in the multi-armed bandit problem (Auer et al. [2002]). In this project, we will adapt a multi-armed bandit method for Online Influence Maximization to the problem of recommendation with serendipity, and analyze the results with respect to accuracy and diversity.

2 Problem of Recommendation With Serendipity

Serendipities are objects that are unexpectedly enjoyable. Two users' reaction to a same recommended item can be poles apart. Thus the difference between serendipity and diversity (Abbassi et al. [2009])should be emphasized: diversity is meant to be strictly about recommended object

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

(dis)similarity (see for instance (Ziegler et al. [2005])), whereas the notion of serendipity also implicitly depends on the user's past behaviour. More formally, diversity can be quantified by the following measure (Vie [2016]):

$$D(V^{(t)}) = \sqrt{|V^{(t)}.t(V^{(t)})|}$$

(where t is the number of rounds, $V^{(t)}$ is the matrix which lines are the feature vectors of explored objects up to time t, and $t(V^{(t)})$ its transpose matrix) (1)

Abbassi et al. [2009] suggest a definition for $\mathcal{OBT}$-ness (*outside-the-box* recommendations), which seems to be an equivalent of the definition of serendipity described above. It relies on a mixture of object and user similarity, and needs offline, fully built user and object similarity networks, along with a few custom parameters which meanings are not straightforward to grasp. Murakami et al. [2007] define a measure of *unexpectedness* which rely on the definition of a baseline method (called "primitive prediction method") to which the accuracy of the method to be evaluated would be compared. Iaquinta et al. [2008] suggest the use of a Naive Bayes classifier on objects (classification as relevant or not with respect to the considered user), although Naive Bayes hypotheses are simplistic, and the algorithm aims at improving diversity in the sense I have described above. Eventually, Kotkov et al. [2016] clearly exhibits the lack of consensus and concise definition of serendipity in recommender systems.

I suggest a definition of serendipity which only relies on:

1. the unweighted, undirected object similarity graph, denoted $\mathcal{G}(V, E)$, which is accessible and computable at start time, because, in the recommender setting, the set of objects is fixed and their feature vectors are known.
2. the past behaviour (in terms of exploration and reward) of the user u the algorithm makes recommendations to, denoted (f_u, r_u) (I assume I never have to cope with a cold start case, which seems relevant with respect to the fact I can still ask the user a few ratings/rewards on some movies in order to "initialize" its behaviour). $f_u^{(t)} : V \to \{0, 1\}$ is a random vector that indicates exploration of an object by user u (at time t) (thus the notion of *support*, that is, the domain on which $f_u^{(t)}$ is positive (denoted Supp($f_u^{(t)}$)), is well-defined). $r_u : V \to \mathcal{R} \subset \mathbb{N}$, a random vector of reward by user u of objects in $\mathcal{G}$. The definition does not assume any prior on the vectors (f_u, r_u), but that I have only access to noisy observations of the rewards and (future) exploration of objects by user u, and that, for any given t, $(f_u^{(t)}, r_u^{(t)})$ only depends on the past behaviour of user u, i.e. on $(f_u^{(0)}, r_u^{(0)}), (f_u^{(1)}, r_u^{(1)}), ..., (f_u^{(t-1)}, r_u^{(t-1)})$. Here, the user features are implicitly described by the distribution of the user behaviour. Note that $(f_u^{(0)}, r_u^{(0)})$ can be set using the starting ratings.

The set of potential serendipities at time $k > 0$ for user u is denoted $\mathcal{S}_u^{(k)}$ and described as follows:

$$\mathcal{S}_u^{(k)} = \underset{v \in \text{Supp}(f_u^{(k)})^c}{\arg\max} \mathbb{E}_{(f_u, r_u)}[\frac{r_u^{(k)}[v] d_e(v, \text{Supp}(f_u^{(k)}))}{\sum_{v' \in \text{Supp}(f_u^{(k)})^c} r_u^{(k)}[v']} | f_u^{(0)}, r_u^{(0)}, ..., f_u^{(k-1)}, r_u^{(k-1)}] \quad (2)$$

where d_e is the distance measure of the length (in number of edges) of the shortest path between two nodes, and, for any subset C of objects, $d_e(., C) = \min_{c \in C} d_e(., c)$, and $\text{Supp}(f_u^{(k)})^c = V - \text{Supp}(f_u^{(k)})$. Intuitively, serendipities can be seen as objects, which are not in already explored regions at time k (i.e. in set $V - \text{Supp}(f_u^{(k)})$), and which give the greatest increase in the "span" of the support of explored regions of the graph, weighted by the known ratings given by the user u. Exploration and rating/reward of a node are drawn from a random distribution: when an object is recommended, the user is "forced" to explore the recommended node, and this might change the exploration/rating distributions (for instance, if the user has enjoyed the recommended item, they will most likely be willing to explore its neighbours). The issue with this definition is that it is not readily computable,

because one does not have access to the random distributions of (f_u, r_u). Nonetheless, sampling methods might be used to evaluate the function in Equation 2. Note that this equation can be seen as merely maximizing some measure of diversity (distance between explored elements) weighted by the reward values. However, one can wonder why I have not used the diversity measure in Equation 1 then. The answer is that, although I deem the diversity measure in Equation 1 interesting to assess the variation in the explored object region, for it is independent of the similarity graph we might build, and only relies on the raw feature vectors, it can be really uninformative in some cases -more specifically, in the case where all unexplored object feature vectors are linearly dependent on the set of explored object feature vectors, because then the value in Equation 2 (provided that we replace d_e by the diversity measure in Equation 1) is equal to zero for any unexplored node.

3 Solving the Problem of Recommendation With Serendipity

Intuitively, what I want is to increase as fast as possible the support of f_u while maximizing the expected rewards of objects in the support. This problem remotely looks like Influence Maximization -I want to select the candidates which will give the highest expected spread, the notion of spread here being the expected reward of unexplored neighbours of each candidate. In order to solve the problem in an online setting, I have adapted the method in Lagrée et al. [2017]. [1] The concept of restricting the set of candidates to a number K of objects (with their "supports", i.e. their direct neighbours in the graph) might suit the idea to restrict recommendations to unexplored suitable regions (see Algorithm 1 for the candidate selection, andAlgorithm 2 for the recommender system). For any matrix A, $A[a, b]$ means the submatrix restricted to indices/elements in sets a (row) and b (column), with possibly a or b reduced to a single element.

Algorithm 1 Candidate selection

Name candidate_selection
Input K: number of candidates in which the recommended item should be selected, s: serendipity threshold, W: (unweighted, undirected) similarity graph matrix, n: total number of objects, f: number of features, F: object feature matrix of size $n \times f$.
Output candidates the set of candidate nodes for recommendation, and supports the set of neighbouring unexplored nodes for each candidate.
- $S \leftarrow \text{Supp}(f_u^{(t)}) \cap \{c \in V : \exists i, 1 \leq i \leq s, W^i[c, \text{Supp}(f_u^{(t)})]\mathbf{1} > 0\}$
- centroids $\leftarrow$ Kmeans(data $= F[S, :]$, nclusters $= K$)
- candidates $\leftarrow \emptyset$
for $c \in$ centroids **do**
  - Append $\arg\min_{v \in S} ||F[v, :] - F[c, :]||_2^2$ to candidates
end for
- supports $\leftarrow \emptyset$
for $v \in$ candidates **do**
  - Append $\{v' \in S : W[v, v'] > 0\}$ to supports
end for

The candidate selection problem is hard. Lagrée et al. [2017] suggest getting the nodes with the highest degree, or using Max-Cover to find plausible candidates. My idea is to implement serendipity at this point of the method: at time t, I apply K-means (*a priori*, any clustering method would be fine, as far as I know) to the set of unexplored elements in the graph at distance lesser than s from the support of $f_u^{(t)}$, and, for each computed centroid, I select the closest unexplored existing node (in terms of L2-norm) to this centroid. I have used the fact that W is an adjacency matrix for an unweighted, undirected graph, thus, for any integer k and coordinates i, j, $W^k[i, j]$ is the number of walks (possibly with repeated edges) of length k between nodes i and j. It is implicitly assumed that the number of centroid nodes will be far lesser than the number of nodes in the object graph.

[1] The idea of seeing the problem of recommendation with serendipity as an influence maximization problem, and adapting the method in Lagrée et al. [2017] is Claire Vernade's. I have designed the adaptation of the method to recommendation with serendipity and implemented it.

Algorithm 2 Adaptation of Lagrée et al. [2017]'s algorithm for Influence Maximization.

Name recommend
Input N: the time budget to evaluate which item should be recommended among the candidates, candidates, supports the output of function candidate_selection.
Output a the recommended item.
- Apply Lagrée et al. [2017]'s method on candidates, supports for N rounds, and compute the spread of each candidate by sampling the reward of each element in its support.
- a is the candidate which maximizes the "candidate score" computed in (Lagrée et al. [2017]) at round N.

4 Experiments

I have compared the adaptation of the method described above with LinUCB (described in Chu et al. [2011]), a completely random strategy (see Algorithm 3), and an ϵ-greedy strategy (see Algorithm 4) on the datasets ml-1m and ml-20m of MovieLens.

First, I have selected all ratings associated with the input username, and restricted the benchmark to the movies rated at least once by the user (to avoid the issue of missing data).

Then, I have computed the object feature matrix F as a one-hot encoding of the movie genres for each object (i.e. were a movie i to be classified as "Romance" and "Comedy", and the available labels be "Romance", "Comedy", "Thriller", "Drama" -in this order- the resulting feature line $F[i,:]$ would be 1100).Then I have built the eps-neighbourhood with parameter σ: first, I have computed the similarity matrix S such that, for two objects i, j, $S(i,j) = \exp(\frac{-||F[i,:]-F[j,:]||_2^2}{\sigma^2})$, and then I have used S to built the adjacency matrix W of the unweighted, undirected similarity graph such that, for two objects i, j:

$$\begin{aligned} W[i,j] &= 1 \text{ if } S(i,j) > \text{eps} \\ &= 0 \text{ otherwise} \end{aligned} \tag{3}$$

I have checked that the provided values for eps and σ^2 had resulted in a connected graph (to avoid having a pair of movies being at infinite distance from each other). The two measures I have used to compare the different bandit models are:

1. The cumulative regret, which is defined as follows at time T:

$$\begin{gathered} R_T = \sum_{t \leq T} \max_{a^* \in V - \text{Supp}(f_u^{(t)})} r(a^*) - r(a^{(t)}) \\ \text{(where } a^{(t)} \text{ is the recommended item at time } t) \end{gathered} \tag{4}$$

   Contrary to $r(a^*)$, $r(a^{(t)})$ is a noisy observation of the true rating:

$$\overline{r(a^{(t)})} \text{ such that } r(a^{(t)}) = \overline{r(a^{(t)})} + \theta \tag{5}$$

   where $\theta \sim \mathcal{N}(0,1)$ (normal distribution of mean 0 and standard deviation 1). When computing the regret for the adapted version of Lagrée et al. [2017]'s method, we also assume that the best arm we could use is among the K selected candidates, thus that we do not challenge the candidate selection method, which is an hypothesis made in the original article.
2. The diversity measure defined in Equation 1. This volume should increase more for recommendation with serendipity than for regular recommendation, because recommended objects are supposed to be less correlated.

Both of these measures are actually averaged across n_iter trajectories. See Figures 6, 7, 8, and 9. It can be noticed that, in all but one figure, the diversity curve converge to zero as the number of

rounds increases. It can be explained by the fact that the total number of movies for user 2 is 129 in dataset ml-1m (for user 1 in ml-20m, it is 175), and at each round a new item is recommended, thus explored. Thus it makes sense that at some point, the feature vectors associated with the explored items might be correlated, thus leading to a zero determinant and a null parallelotope volume. This might account for the fact that, although the ϵ-greedy strategy should maximize the diversity measure in Equation 1, it actually performs poorly, because at some point (especially in this dataset where there is only relatively a few values of possible feature vectors), any unexplored object has a feature vector (linearly) dependent with the set of explored object feature vectors, thus the diversity values are all zeroes, and then the strategy boils down to choosing the first object in the list of unexplored ones. From the regret curve, as expected, LinUCB is the most accurate of the methods, but performs poorly with respect to the diversity measure, which is constant. It is due to the fact that once it finds the best arm, it repeatedly plays it. The adapted method has a quasi-linear regret curve, although it performs better than the other methods with respect to the diversity measure. Figure 10 compares the performance of the method for varying values of the serendipity threshold s. According to this figure, the algorithm performs best for $s = 4$ both in terms of diversity and regret. The evolution of the regret (resp. diversity) curve depending on the value of s cannot straightforwardly infered; it might need a benchmark on another dataset, or a higher value of n_iter to be able to draw conclusions from the figures. Note that the horizon cannot exceed ≈ 120, because the total number of items in the ml-1m subset of MovieLens for user denoted $u = 2$ is around this number.

Since the definition of serendipity 2 cannot be computed directly (because we do not know the distribution of (f_u, r_u)), we will use a Monte-Carlo estimate (i.e. the empirical mean) to assess the evolution of the function associated with serendipities.

Algorithm 3 Random strategy.

Name recommend_random
Input n: number of objects.
Output a the recommended item.
- $a \sim \mathcal{U}(\{1, 2, ..., n-1, n\})$ (uniform distribution on a discrete set of size n).

Algorithm 4 ϵ-greedy strategy.

Name recommend_greedy
Input n: number of objects, ϵ the probability of playing the random strategy.
Output a the recommended item.
- Draw $e \sim \mathcal{B}(\epsilon)$ (Bernouilli distribution of parameter ϵ).
if $e = 1$ **then**
  - $a \sim \mathcal{U}(\{1, 2, ..., n-1, n\})$
else
  - $a = \arg\max_{b \in V - \text{Supp}(f_u^{(t)})} D(V^{t-1}, F[b, :])$ (diversity measure where the feature vector of b is added as the t^{th} line of the observed feature matrix).
end if

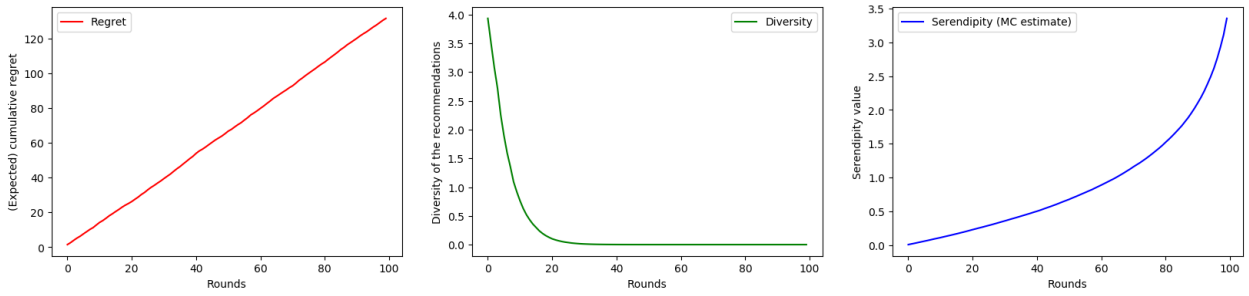

Figure 1: [Random strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

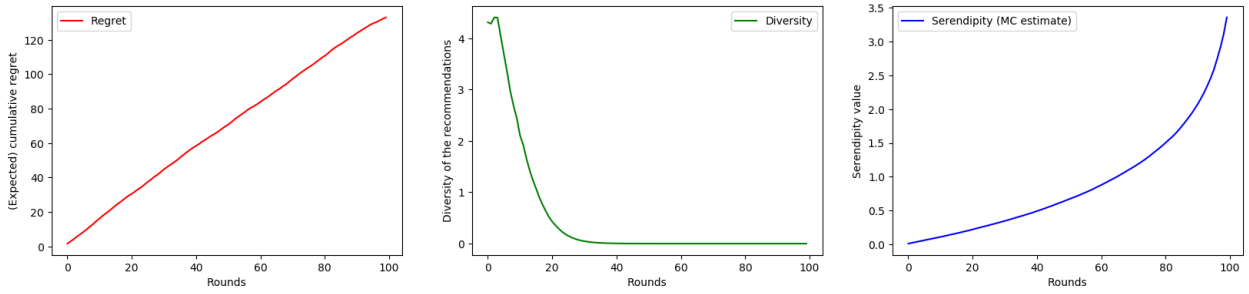

Figure 2: [ϵ-greedy strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $\epsilon = 0.2$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

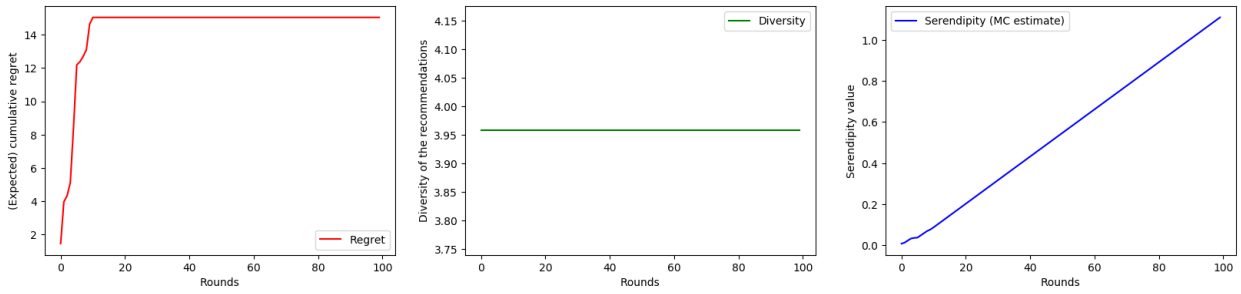

Figure 3: [LinUCB strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $\alpha = 0.1$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

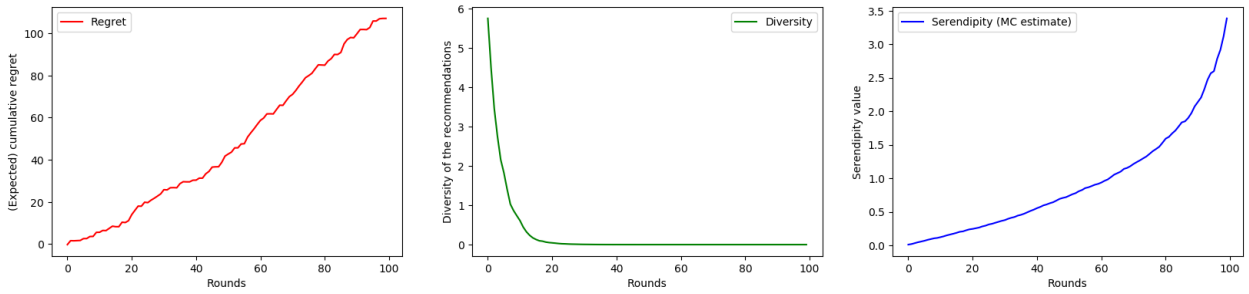

Figure 4: [Adapted method strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $K = 5, s = 1, N = 10$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

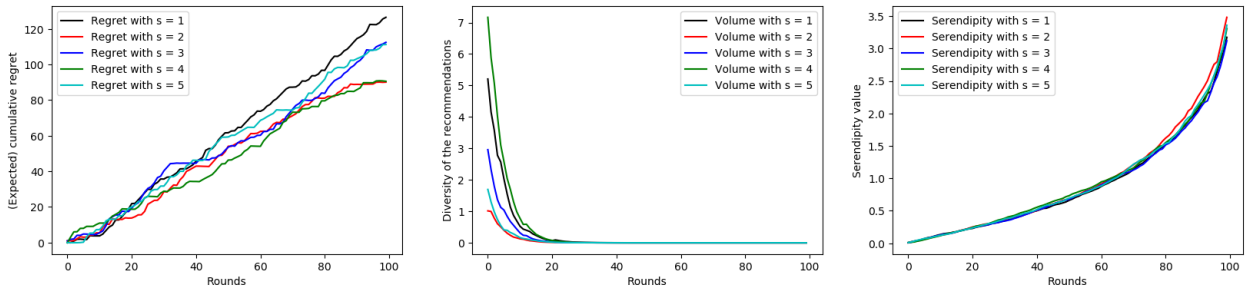

Figure 5: [Adapted method strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $K = 5, N = 10$ with varying parameter $s = 1, 2, 3, 4, 5$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

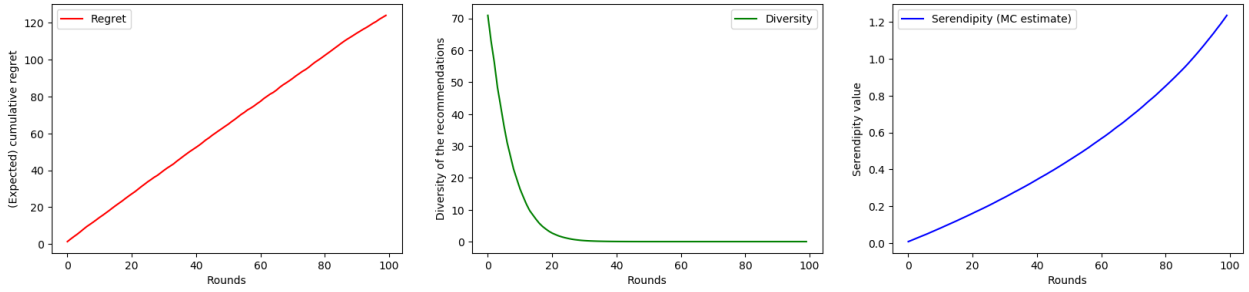

Figure 6: [Random strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 1 in MovieLens dataset ml-20m. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

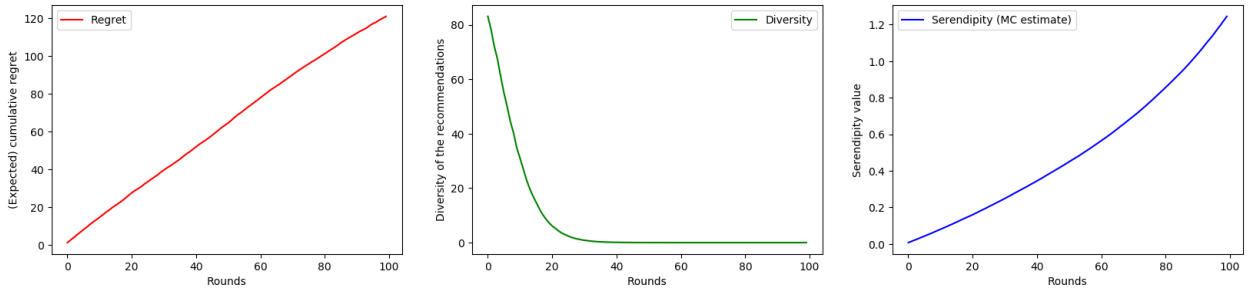

Figure 7: [ϵ-greedy strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 1 in MovieLens dataset ml-20m and parameter $\epsilon = 0.2$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

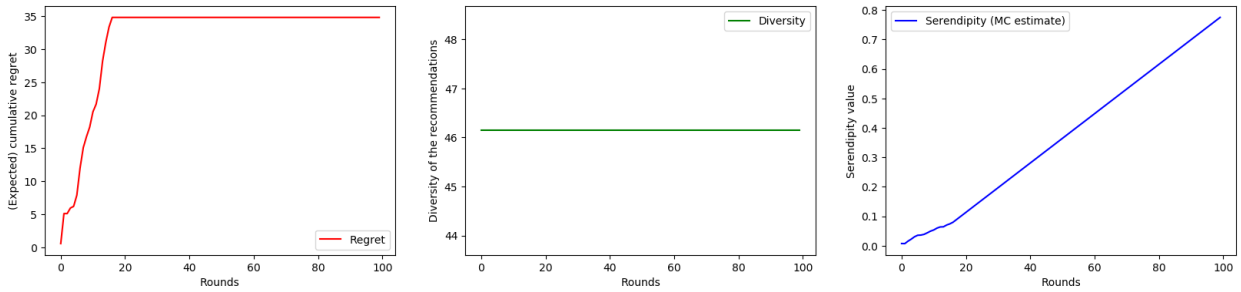

Figure 8: [LinUCB strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 1 in MovieLens dataset ml-20m and parameter $\alpha = 0.1$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

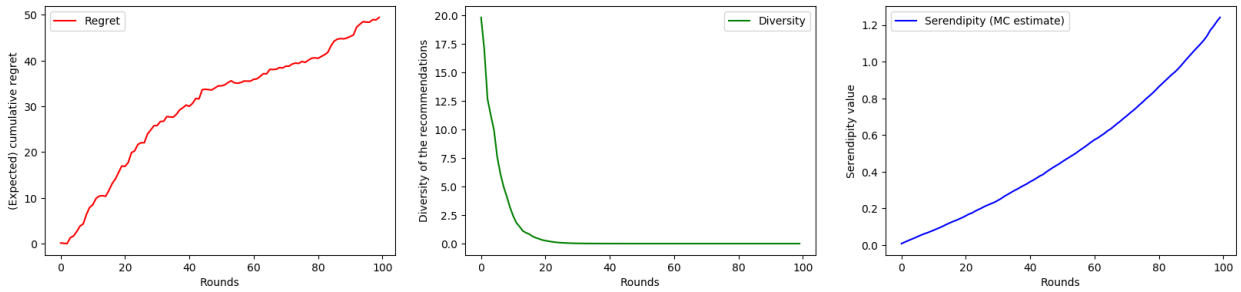

Figure 9: [Adapted method strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 1 in MovieLens dataset ml-20m and parameter $K = 5, s = 1, N = 10$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

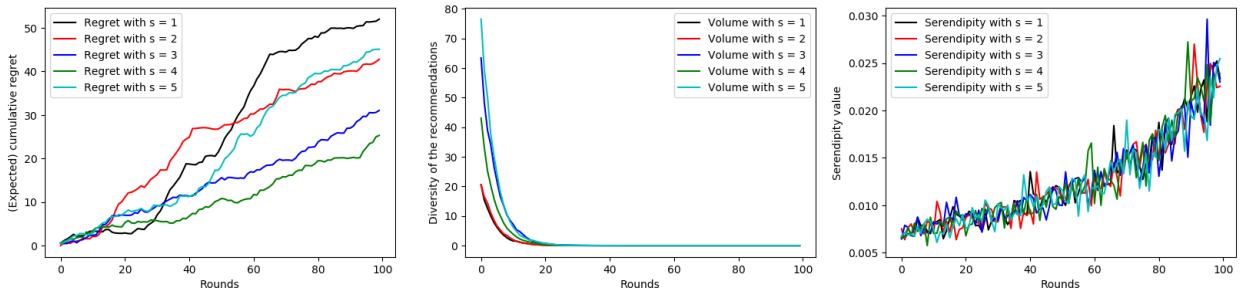

Figure 10: [Adapted method strategy] (Average) cumulative regret, diversity measure, and empirical cumulative serendipity value (maximized function in Equation 2) over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 1 in MovieLens dataset ml-20m and parameter $K = 5, N = 10$ with varying parameter $s = 1, 2, 3, 4, 5$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

Method	Average run time in dataset ml-1m	Average run time in dataset ml-20m
random	32 sec.	48 sec.
ϵ-greedy	1 min. 30 sec.	2 min. 16 sec.
LinUCB	39 sec.	53 sec.
Adapted method	1 min. 31 sec.	1 min. 49 sec.

Table 1: Average run times (over $n\_iter = 100$) with horizon $T = 100$.

Note that the runtimes listed in Table 4 use Python implementations of each bandit model; coding them in C++ instead would have probably made everything faster. The code itself is not really optimized either, and this may account for the slowness of ϵ-greedy and the adapted method.

5 Discussion

TODO

1. Rotting Bandits
2. Abbassi et al. [2009]'s method
3. Using similarity between users?
4. There might be a way to formalize the equation about serendipity in terms of (object similarity) (possibly random walk) Graph Laplacian (up to reweighting using the reward observations), thus using the classic methods and potentially inheriting of interesting properties.

References

Zeinab Abbassi, Sihem Amer-Yahia, Laks VS Lakshmanan, Sergei Vassilvitskii, and Cong Yu. Getting recommender systems to think outside the box. In *Proceedings of the third ACM conference on Recommender systems*, pages 285–288. ACM, 2009.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6):734–749, 2005.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Keith Bradley and Barry Smyth. Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, pages 85–94. Citeseer, 2001.

Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

Leo Iaquinta, Marco De Gemmis, Pasquale Lops, Giovanni Semeraro, Michele Filannino, and Piero Molino. Introducing serendipity in a content-based recommender system. In *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*, pages 168–173. IEEE, 2008.

Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111:180–192, 2016.

Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-Based Systems*, 123:154–162, 2017.

Paul Lagrée, Olivier Cappé, Bogdan Cautis, and Silviu Maniu. Effective large-scale online influence maximization. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 937–942. IEEE, 2017.

Tomoko Murakami, Koichiro Mori, and Ryohei Orihara. Metrics for evaluating the serendipity of recommendation lists. In *Annual conference of the Japanese society for artificial intelligence*, pages 40–46. Springer, 2007.

Brent Smith and Greg Linden. Two decades of recommender systems at amazon. com. *Ieee internet computing*, 21(3):12–18, 2017.

Tiffany Ya Tang and Gordon McCalla. Smart recommendation for an evolving e-learning system. In *Workshop on Technologies for Electronic Documents for Supporting Learning, International Conference on Artificial Intelligence in Education*, pages 699–710, 2003.

Jill-Jênn Vie. *Modèles de tests adaptatifs pour le diagnostic de connaissances dans un cadre d'apprentissage à grande échelle*. PhD thesis, Université Paris-Saclay, 2016.

Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.