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 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 dissimilarity, 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]):

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

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

Abbassi et al. [2009] suggests 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.

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, for in the recommender setting the set of objects is fixed and their feature vectors are known.
2. a *serendipity threshold*, denoted s, which quantifies the risk run in order to surprise the user (the higher it is, the more regions of the object graph are available for recommendation).
3. 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). $f_u^{(t)} : V \rightarrow \{0, 1\}$ 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). $r_u : V \rightarrow \mathcal{R} \subset \mathbb{N}$, 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 t for user u (with threshold s) is denoted $\mathcal{S}_{u,s}^{(t)}$ and described as follows:

$$\mathcal{S}_{u,s}^{(t)} = \underset{\substack{v \in V - \text{Supp}(f_u^{(t)}) \\ d(v, \text{Supp}(f_u^{(t)})) \leq s}}{\arg\max} \mathbb{E}_{(f_u, r_u)}[\sum_{v' \in V} [f_u^{t+1}(v") - f_t - u(v")] r_u(v")] \tag{2}$$

where , , d is the distance measure used to build G and, for any subset C of objects, $d(., C) = \min_{c \in C} d(., c)$, and s the so-called "serendipity threshold".

Intuitively, they can be seen as objects, which are not in already explored regions at time t (i.e. in set $V -$ Supp($f_u^{(t)}$)), and which give the greatest increase in the diameter/support of explored regions of the graph, weighted by the known ratings given by the user u.

No assumptions made on past behaviour and ratings. But not computable. More general than the previous definition. ratings may depend on user similarity!!

where E is the expectation: exploration and rating of a node are drawn from a random distribution: when an object is recommended, we "make" the user explore the recommended node, and this modifies the exploration/rating distributions (if the user has enjoyed the recommended object, they will most likely be willing to explore the neighbours of this object). f-u would likely be submodular (a node can be explored only once, thus at a given point in time, the number of explored elements will increase less and less strongly).

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.

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 $N$.
```

[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.

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 dataset ml-1m 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, $S(i,j) = \text{exp}(\frac{-||F[i,:]-F[j,:]||_2^2}{\sigma^2})$, 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:

$$R_T = \tag{4}$$

   When computing the regret, we also assume that the best "arm" we could use is among the K selected candidates, thus that we do not question the method with which we select the candidates, which is an hypothesis made in the [Lagrée et al., 2017] article.

2. 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.

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, $\epsilon$ the probability of playing the random strategy.
Output $a$ the recommended item.
- Draw $e \sim \mathcal{B}(\epsilon)$ (Bernouilli distribution of parameter $\epsilon$).
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
```

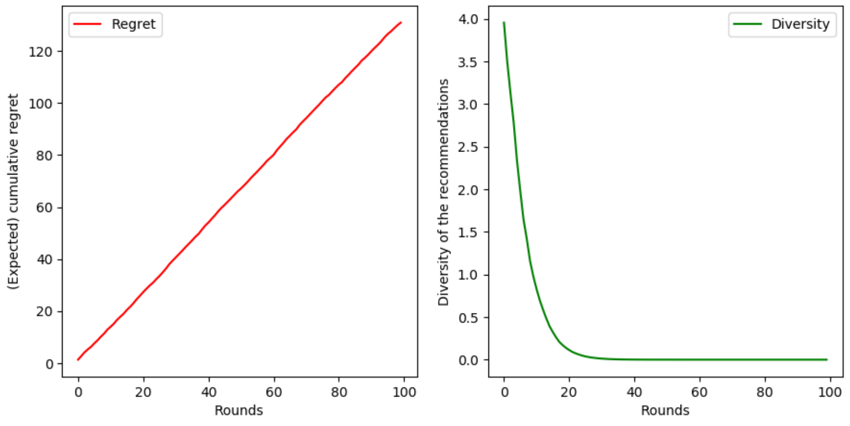

Figure 1: [Random strategy] (Average) cumulative regret and diversity measure over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 = 100$.

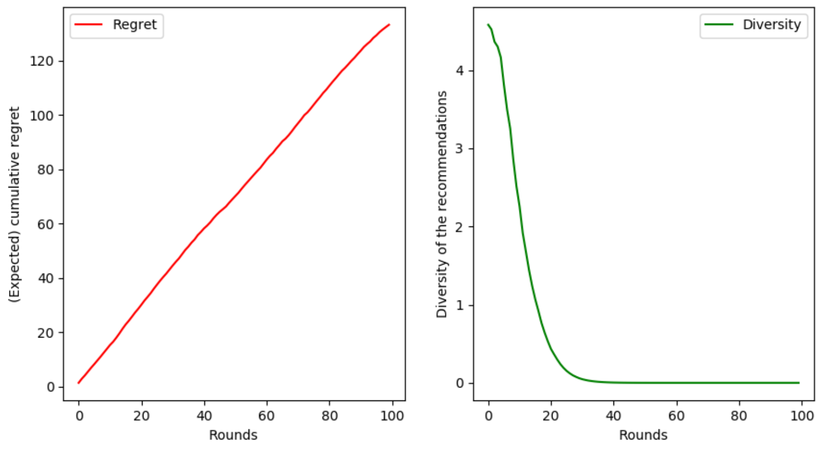

Figure 2: [ϵ-greedy strategy] (Average) cumulative regret and diversity measure over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $\epsilon = 0.2$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2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

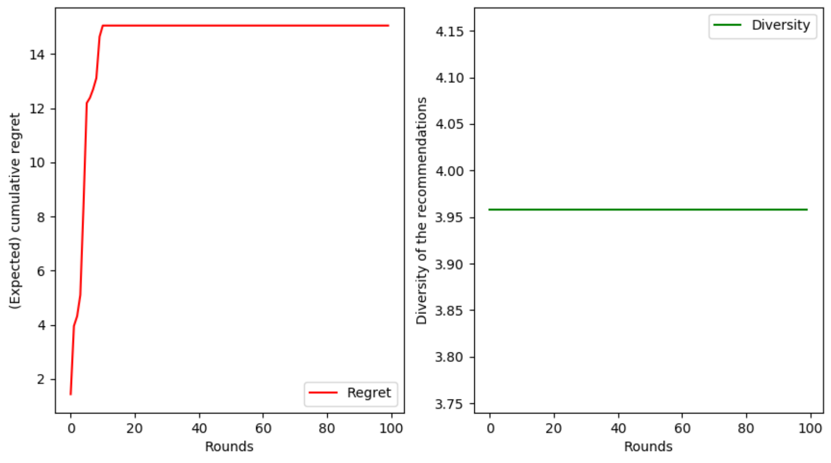

Figure 3: [LinUCB strategy] (Average) cumulative regret and diversity measure over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $\alpha = 0.1$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$, $\sigma^2 =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

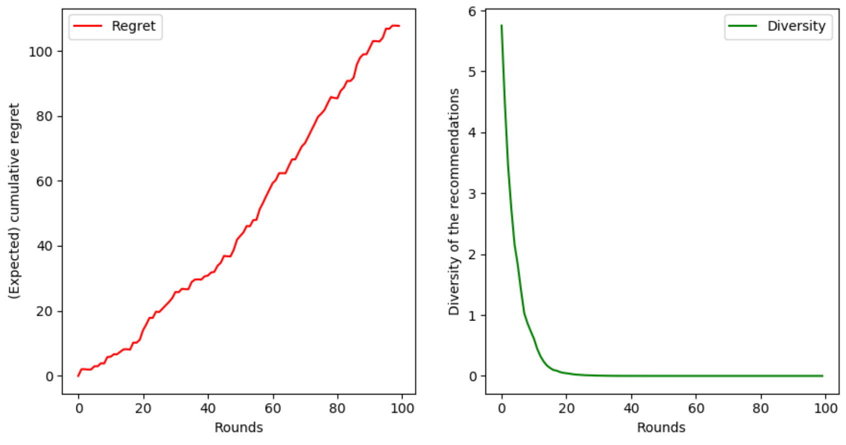

Figure 4: [Adapted method strategy] (Average) cumulative regret and diversity measure over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $K = 5, s = 1, N = 10$. In order to build the eps-neighbourhood similarity graph, I have used values eps $= 0.6$,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

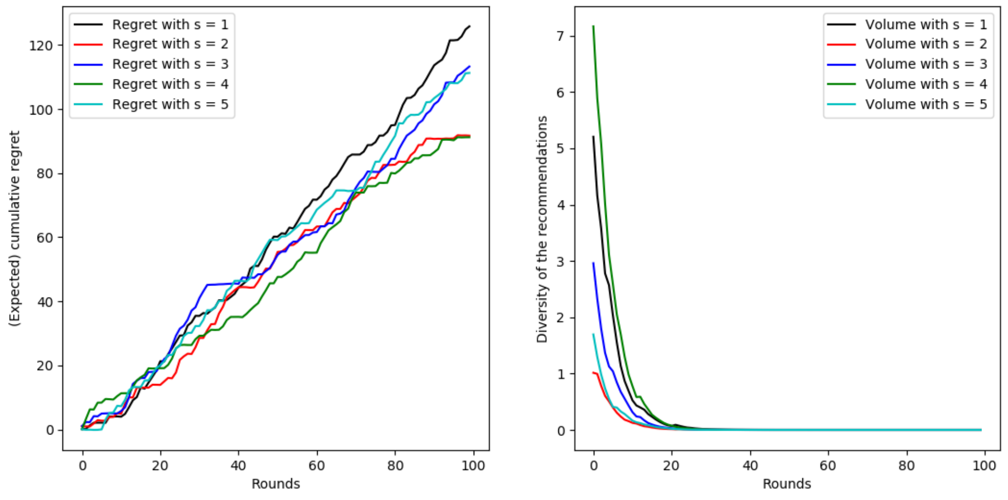

Figure 5: [Adapted method strategy] (Average) cumulative regret and diversity measure over $n\_iter = 100$ trajectories, horizon $T = 100$, for user 2 in MovieLens dataset ml-1m and parameter $K = 5, N = 10$ with varying parameter $s = 1, 2, 3, 4, 5$. In order to build the eps-neighbourhood similarity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

Table 1: Run times for $n\_iter = 100$, horizon $T = 100$ for each method.

Method	Run time
random	4 min. 40 sec.
ϵ-greedy	51 min. 6 sec.
LinUCB	6 min. 4 sec.
Adapted method	36 min. 16 sec.

Note that the runtimes listed in Table 1 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

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.

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.

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.