Topic-Level Bayesian Surprise and Serendipity for Recommender Systems

Tonmoy Hasan*
thasan1@uncc.edu
University of North Carolina at Charlotte
Charlotte, North Carolina, USA

Razvan Bunescu*
razvan.bunescu@uncc.edu
University of North Carolina at Charlotte
Charlotte, North Carolina, USA

ABSTRACT

A recommender system that optimizes its recommendations solely to fit a user's history of ratings for consumed items can create a filter bubble, wherein the user does not get to experience items from novel, unseen categories. One approach to mitigate this undesired behavior is to recommend items with high potential for serendipity, namely surprising items that are likely to be highly rated. In this paper, we propose a content-based formulation of serendipity that is rooted in Bayesian surprise and use it to measure the serendipity of items after they are consumed and rated by the user. When coupled with a collaborative-filtering component that identifies similar users, this enables recommending items with high potential for serendipity. To facilitate the evaluation of topic-level models for surprise and serendipity, we introduce a dataset of book reading histories extracted from Goodreads, containing over 26 thousand users and close to 1.3 million books, where we manually annotate 449 books read by 4 users in terms of their time-dependent, topic-level surprise. Experimental evaluations show that models that use Bayesian surprise correlate much better with the manual annotations of topic-level surprise than distance-based heuristics, and also obtain better serendipitous item recommendation performance.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Natural language processing*; **Online learning settings**.

KEYWORDS

surprise and serendipity, non-stationary time series, topic distributions.

ACM Reference Format:
Tonmoy Hasan and Razvan Bunescu. 2023. Topic-Level Bayesian Surprise and Serendipity for Recommender Systems. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3604915.3608851

*Hasan contributed mainly the dataset processing, the manual annotation, the implementation, tuning, and evaluation of the proposed methods. Bunescu contributed mainly the task formulations, the algorithms, the formal definitions of surprise and serendipity, the overall study design, and the writing.

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09…$15.00
https://doi.org/10.1145/3604915.3608851

1 INTRODUCTION

Recommender systems offer personalized, ranked lists of user-relevant items, easing the cognitive overload caused by the ever growing number of items available to users. Adapting techniques from information retrieval, machine learning, and data mining, recommender systems have a long and rich history, ranging from classical approaches [31, 32] to modern methods based on advances in deep learning and dense representations of users and items [15, 38]. Traditionally, recommender systems optimize their output using content-based (CB) signals and collaborative filtering (CF) information, either separately or in combination [26]. Content-based approaches recommend items that are similar to previously consumed liked items, whereas collaborative filtering methods seek to find items that were liked by similar users. By constantly optimizing for these two objectives, recommender systems can create *filter bubbles* [11, 30] where a user is insulated from topics or points of view that are different from the ones they have already been exposed to. This phenomenon is further exacerbated by the feedback loop between the item ranking model and the passive user reaction to the recommended items [24], where the ⟨item, rating⟩ pair is used as a new sample to update the ranking model, thus reinforcing historical user preferences. A widely used concept for alleviating filter bubbles is that of diversity, where recommender systems are encouraged to generate recommendation sets containing dissimilar items [8]. Increasing diversity can be done by post-hoc reranking of recommended items, using approaches such as maximal marginal relevance [6, 40]. Additionally, diversity measures can be incorporated alongside relevance criteria in the objective function and used during training and inference [13, 37, 39], inter alia. All these methods alleviate the filter bubble by seeking to model and optimize for diversity directly. An alternative strategy is to optimize for measures that have an indirect, but strong positive effect on diversity. Such a measure is serendipity, where the aim is to recommend items that surprise the user in a positive way [1, 2, 7, 19, 21, 23, 27, 29, 33]. In this paper, we introduce a formal characterization of topic-level serendipity based on Bayesian surprise, and evaluate its various implementations on a new dataset of book reviews that have been manually annotated with surprise labels.

Reflecting a definition of serendipity as the occurrence of an event that is both *surprising* and *valuable*, in Sections 2 and 3 we introduce a content-based, post-factum measure of serendipity that uses Bayesian surprise [17] and user ratings to capture the two definitional aspects of serendipity. However, by itself, the serendipity

component can estimate serendipity only after an item has been consumed and rated. Subsequently, recommendation of serendipitous items is enabled by integrating the measurement of topic-level serendipity with a collaborative filtering component that is tasked with identifying similar users who consumed serendipitous items in the past. Computing Bayesian surprise requires maintaining a probability distribution over user preferences and updating it every time an item is consumed and rated. Correspondingly, in Section 3 we describe a number of online learning algorithms that were adapted to work in a non-stationary setting where user preferences can drift over time. These are supplemented in Section 4 with methods that estimate surprise using distances between point estimates of preference vectors at consecutive times steps in the user's time series of consumed items. For evaluation, in Section 5 we introduce a dataset derived from Goodreads [36] in which users are associated time series of books and ratings, and where books are manually annotated as to whether they present topic-level surprise at the time they appeared in the user's time series. The experiments in Section 6 show that methods that rely on Bayesian surprise are better at predicting topic-level serendipity.

2 TASK DEFINITION

We assume there is a set of books, or more generally a set of **items** $\mathcal{I}$. There is also a set of K **topics**, and for each book item $i \in \mathcal{I}$ there is a corresponding **topic distribution** $\boldsymbol{\theta}_i = [\theta_{i,1}, \theta_{i,2}, ..., \theta_{i,K}]$, where $\sum_k \theta_{i,k} = 1$. Furthermore, there is a set of **users** $\mathcal{U}$ who consume items over time. For a given user $u \in \mathcal{U}$, let the vector $\mathbf{p}_t(u) = [p_{t,1}, p_{t,2}, ..., p_{t,K}]$ capture his set of **user preferences** over the K topics at time t, where $p_{t,k}$ represents the user's preference for topic k at time t (the argument u was left implicit). User preferences are unbounded and can be positive or negative, corresponding to the user liking or disliking that topic, respectively. At a time step t, upon consuming item $\boldsymbol{\theta}_i$, the user experiences a **reward** r_t, which expresses how much he[1] liked (positive reward) or disliked (negative reward) that item. Henceforth, to simplify notation, we use $\boldsymbol{\theta}_i$ to denote either item number i or the item consumed by the user at time step i, where the user is evident from the context. To recommend items with high serendipity, we rely on the following two components:

(1) **Item Surprise**: Estimate the amount of surprise $Sur(u, t)$ that the book θ_t generated for a user u who has consumed and rated the books $\langle\theta_1, r_1\rangle$, ..., $\langle\theta_{t-1}, r_{t-1}\rangle$, and $\langle\theta_t, r_t\rangle$, in this order.

(2) **User Similarity**: Given user u at time step i and another user v at time step j, estimate how different the two users are in terms of their preferences, as a distance $d(\langle u, i\rangle, \langle v, j\rangle)$.

Given the two components above, recommending serendipitous items to a user u at a time step $i+1$ will be done in a collaborative filtering manner by identifying users v who at a time step j in their past are similar in terms of their preferences to user u and furthermore who, at their next time step $j+1$, consumed an item that was *positively* rated and that also resulted in a high level of *surprise*, hence *serendipitous*. This recommendation procedure is detailed in Algorithm 1, as follows. In line 1, the algorithm searches in the entire database of users v and their time series of consumed items j to find the N users whose preferences upon consuming item j were most similar with user u's preferences upon consuming item i. This set is further filtered in line 2 to keep only those users whose similarity is not below a predefined threshold and who rated positively the next item. Of these users, line 3 identifies the one whose next item resulted in the largest amount of surprise, which is returned in line 4.

Algorithm 1: FindSerendipity(u, i)

Input: A reference user u and a time step i.
The preference distance threshold τ_d (hyperparameter).
The number N of most similar users to consider (hyperparameter).

Output: A pair $\langle v, j\rangle$ similar to $\langle u, i\rangle$ where item $j+1$ has high serendipity for user v.

1 Let T = the top set of N pairs $\langle v, j\rangle$ most similar to $\langle u, i\rangle$; /* N lowest $d(\langle u, i\rangle, \langle v, j\rangle)$ */

2 Let $S = \{\langle v, j\rangle \in T \mid d(\langle u, i\rangle, \langle v, j\rangle) < \tau_d \ \wedge r_{j+1} > 0\}$; /* high similarity and positive rating */

3 Let $\langle v, j\rangle = \underset{\langle v,j\rangle \in S,\ v \neq u}{\arg\max}\ Sur(v, j+1)$; /* next item with maximum surprise */

4 **return** $\langle v, j\rangle$; /* return null pair if S is empty */

To instantiate the algorithm above, we need to specify how to compute the topic-level surprise measure $Sur(u, t)$ and the user preference distance $d(\langle u, i\rangle, \langle v, j\rangle)$. To estimate the surprise $Sur(u, t)$ we investigate two types of approaches:

- **Online learning**: In this approach (Section 3), the preference vector $\mathbf{p}_t(u)$ is updated at every step to minimize the rating loss $(\hat{r}_t - r_t)^2$ incurred from predicting a rating $\hat{r}_t$ at that time step. We consider linear reward estimation models $\hat{r}_t = \mathbf{p}^T \boldsymbol{\theta}_t$, where $\mathbf{p}$ can be seen as the parameters and $\boldsymbol{\theta}_t$ the input features at time t. Bayesian surprise $Sur(u, \boldsymbol{\theta}_t)$ will then be calculated as the KL divergence between the preference distributions at times t and $t-1$, whereas
- **Basic model**: In this approach (Section 4), simple weighted averages of topic distributions and heuristic distances to compute preference vectors $\mathbf{p}_t(u)$ and surprise values $Sur(u, t)$, respectively.

In all approaches, the preference distance $d(\langle u, i\rangle, \langle v, j\rangle)$ will be computed simply as the L2 norm of the difference between the two preference vectors $\mathbf{p}_i(u)$ and $\mathbf{p}_j(v)$, i.e., $d(\langle u, i\rangle, \langle v, j\rangle) = \left\|\mathbf{p}_i(u) - \mathbf{p}_j(v)\right\|_2$.

In the Goodreads dataset, readers provide ratings that are on a scale of 1 to 5 stars. These ratings are projected onto the $[-2, 2]$ interval by subtracting 3 stars from the raw star values (henceforth, we use the terms reward and rating interchangeably). Using the linear reward estimation model, training sequential learning models to fit a label in $[-2, 2]$ is approached as an online linear regression (LR) task. In this context, we define the post-factum **serendipity** of item θ_t for user u as the product between the rating r_t and the

[1]The user gender was sampled at random by tossing a coin.

surprise $Sur(u,t)$ that θ_t triggered for u:

$$Ser(u,t) = r_t \times Sur(u,t) \tag{1}$$

Given that all surprise measures described in Section 3 below are positive and a positive reward corresponds to a raw rating strictly greater than 3 stars, the definition above maps well to the conceptual definition of serendipity as an event that is both surprising, i.e., large $Sur(u,t)$, and valuable, i.e., $r_t > 0$. Looking back at Algorithm 1, by only selecting items $j+1$ with positive rating (line 3) that maximize surprise (line 4), the procedure can be seen as returning an item with high serendipity value for v. Since v itself was selected to have similar preferences with user u, the collaborative filtering expectation is that item $j+1$ will be serendipitous for user u as well.

3 ONLINE LEARNING OF NON-STATIONARY USER PREFERENCES

We assume that at each time step t a user's preference vector has a multivariate normal distribution $\mathbf{p}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$. In this section, we describe online learning algorithms that take as input the preference distribution at the previous time step $\mathbf{p}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_{t-1}, \Sigma_{t-1})$, the topic vector for the current item $\boldsymbol{\theta}_t$, and the user's rating r_t, and update the preference distribution to $\mathbf{p}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma_t)$ in order to minimize the squared distance between the predicted rating $\hat{r}_t$ and the true rating r_t. Under a Bayesian interpretation, $\mathbf{p}_{t-1}$ is the prior preference distribution whereas $\mathbf{p}_t$ is the posterior distribution upon observing item $\boldsymbol{\theta}_t$ and rating r_t. Bayes rules is indeed how the posterior distribution is derived in the Bayesian linear regression approach described in Section 3.1 below.

Under the framework of Bayesian surprise of Itti and Baldi [17], the **surprise** elicited by the item $\boldsymbol{\theta}_t$ from user u is defined as the distance between the posterior and prior distributions, measured using KL divergence [20]:

$$Sur(u,t) = KL\left(\mathcal{N}(\mu_t, \Sigma_t) || \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})\right) \tag{2}$$

In Sections 3.1 and 3.2 we introduce adaptations of Bayesian linear regression and adaptive regularization of weights (AROW) for the estimation of non-stationary user preference distributions, which will enable calculating Bayesian surprise as defined above. Note that these methods will require the rating r_t to compute the posterior distribution, which means surprise will be measured post-factum, and may be caused by both novel topic distributions and unexpected ratings (a different definition using only the covariance matrices is left for future work). A post-factum definition of surprise is also used in the NLMS approach from Section 3.3.

3.1 Variance Bounded Bayesian Linear Regression

The first sequential learning model that we consider for updating the user preference distribution is that of Bayesian linear regression [4]. Under a Bayesian treatment of linear regression, at the beginning $t = 0$ of a user's reading history, its preference vector is distributed according to a zero-centered Gaussian prior $\mathbf{p}_0 \sim \mathcal{N}(\mathbf{0}, \beta I)$. Due to the choice of a conjugate Gaussian prior distribution, it can be shown that the posterior at the next time step will be Gaussian as well according to the following update rules (section 3.3 in [4]):

$$\Sigma_{t+1}^{-1} = \Sigma_t^{-1} + \beta \boldsymbol{\theta}_t^T \boldsymbol{\theta}_t \quad \text{and} \quad \boldsymbol{\mu}_{t+1} = \Sigma_{t+1}\left(\Sigma_t^{-1}\boldsymbol{\mu}_t + \beta \boldsymbol{\theta}_t^T r_t\right) \tag{3}$$

The original Bayesian LR model is intended for a *stationary* setting where the true model that generates the data and their labels does not change over time. However, in a recommendation setting, users change their preferences over time, which requires the LR model to be flexible enough to allow for sometimes drastic changes in its parameters, depending on how much the true user preferences changed. Using the original Bayesian LR model in such a *non-stationary* setting is then going to be suboptimal due to the fact that once the parameter co-variance Σ gets close to 0, the parameters change very little. To alleviate this issue, we introduce *variance bounded Bayesian LR* that ensures the variance of each parameter is at least τ_v, where $\tau_v > 0$ is a hyperparameter. At every update step, we take the current covariance matrix Σ_t, perform an eigenvalue decomposition $\Sigma_t = U_t \Lambda U_t^T$, where $\Lambda = diag(\lambda_1, ..., \lambda_K)$ is the diagonal matrix of positive eigenvalues. Then all eigenvalues are clipped to be at least τ_v and used in a new diagonal matrix of clipped eigenvalues $\Lambda_v = clip(\Lambda, \tau_v) = diag(max(\lambda_1, \tau_v), ..., max(\lambda_K, \tau_v))$. We use Λ_v to compute a new covariance matrix $S_t = U_t \Lambda_v U_t^T$, which is then employed as usual in the Bayesian LR update equations, as shown below.

$$\Sigma_t = U_t \Lambda U_t^T \longrightarrow \Lambda_v = clip(\Lambda, \tau_v) \longrightarrow S_t = U_t \Lambda_v U_t^T \tag{4}$$

$$\Sigma_{t+1}^{-1} = S_t^{-1} + \beta \boldsymbol{\theta}_t^T \boldsymbol{\theta}_t \quad \text{and} \quad \boldsymbol{\mu}_{t+1} = \Sigma_{t+1}\left(S_t^{-1}\boldsymbol{\mu}_t + \beta \boldsymbol{\theta}_t^T r_t\right) \tag{5}$$

3.2 Adaptive Regularization of Weights for Regression

The Adaptive Regularization of Weights (AROW) algorithm [9] is an online optimization procedure for confidence-weighted learning of linear classifiers [10]. The original formulation of AROW is:

$$\mu_t, \Sigma_t = \arg\min_{\mu,\Sigma} C(\mu, \Sigma) \tag{6}$$

where the objective function $C(\mu, \Sigma)$ is written as:

$$KL\left(\mathcal{N}(\mu, \Sigma) || \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})\right) + \frac{l(r_t, \mu^T \theta_t)}{2r_1} + \frac{\theta_t^T \Sigma \theta_t}{2r_2} \tag{7}$$

The two hyperparameters r_1 and r_2 quantify the trade-off between the prediction accuracy and the confidence in the new parameters. By minimizing the first term (Bayesian surprise), the new parameters are encouraged to stay close to the current values. In the original AROW, the loss l was set to be the squared hinge loss for classification.

We adapt AROW for regression by replacing the original squared-hinge loss with the squared error loss $l(r_t, \mu^T \theta_t) = (r_t - \mu^T \theta_t)^2$. Furthermore, whereas the hyperparameters r_1 and r_2 were set to be equal in [9], in our experiments they are tuned separately. In a non-stationary setting it is especially important that r_2 is independent from r_1, to allow the parameters to vary more freely when the user changes his preferences.

To solve for the parameters μ, Σ that minimize equation (1), we use a derivation analogous to [9] by writing the KL term explicitly and decomposing the loss in two parts depending on μ and Σ, i.e.,

$C(\mu, \Sigma) = C_1(\mu) + C_2(\Sigma)$:

$$C_1(\mu) = \frac{1}{2}(\mu_{t-1} - \mu)^T \Sigma_{t-1}^{-1}(\mu_{t-1} - \mu) + \frac{1}{2r_1}(r_t - \mu_t^T x_t)^2 \quad (8)$$

$$C_2(\Sigma) = \frac{1}{2}\log\frac{det\,\Sigma_{t-1}}{det\,\Sigma} + \frac{1}{2}Tr(\Sigma_{t-1}^{-1}\Sigma) + \frac{1}{2r_2}x_t^T \Sigma x_t \quad (9)$$

By settings the gradients of Equations 8 and 9 to zero, we obtain the following online update equations:

$$\mu_t = \mu_{t-1} + \frac{(r_t - \mu_{t-1}^T x_t)\Sigma_{t-1}x_t}{r_1 + x_t^T \Sigma_{t-1} x_t} \quad (10)$$

$$\Sigma_t = \Sigma_{t-1} - \frac{\Sigma_{t-1}x_t x_t^T \Sigma_{t-1}}{r_2 + x_t^T \Sigma_{t-1} x_t} \quad (11)$$

Note that by keeping r_1 and r_2 separate, this formulation is different from the AROW version in [35].

3.3 Normalized Least Mean Square

We also experiment with normalized least mean square (NLMS) [3], a more stable version of the well known least mean square (LMS) algorithm for regression, where the learning rate is divided by the squared norm of the feature vector:

$$\boldsymbol{p}_t = \boldsymbol{p}_{t-1} + \frac{\eta}{\boldsymbol{\theta}_t^T \boldsymbol{\theta}_t}(r_t - \boldsymbol{p}_{t-1}^T \boldsymbol{\theta}_t)\boldsymbol{\theta}_t \quad (12)$$

$$Sur(v, t) = ||\boldsymbol{p}_{t+k-1} - \boldsymbol{p}_{t-1}|| \quad (13)$$

Note that NLMS only updates a point estimate $\boldsymbol{p}_t = \boldsymbol{\mu}_t$ of the preference vector, therefore it cannot be used to compute Bayesian surprise. Instead, we define the surprise at time t as the the norm of the difference between the preference vectors at times $t-1$ and $t+k-1$, where $k \geq 1$ is a time horizon hyper-parameter. When the time horizon is given the default value $k = 1$, this measure can be seen as capturing the impact that the item θ_t had on the preference vector update. Larger values of k are meant to model the fact that the impact of an item item θ_t may become clearer only after multiple NLMS updates. While it may appear that preference vectors from the "future" are used, this future is only relative to a point t in the past for a user v and it is not an issue as long as it is done only for finding similar users v and their post-factum surprising items in the collaborative filtering step from Algorithm 1 (no future information is used for the reference users u during evaluation).

4 BASIC MODEL FOR SURPRISE AND USER PREFERENCES

A simpler approach is to decouple the rating from surprise and define surprise as a distance d between an item $\boldsymbol{\theta}_t$ and the previously consumed items summarized in a topic history vector $\boldsymbol{h}_{t-1}$:

$$Sur(u, t) = d(\boldsymbol{\theta}_t, \boldsymbol{h}_{t-1}) \quad \text{where} \quad \boldsymbol{h}_{t-1} = \frac{1}{t-1}\sum_{z=1}^{t-1}\boldsymbol{\theta}_z \quad (14)$$

Here, $\boldsymbol{h}_{t-1}$ embeds the history of items into an average topic distribution vector aimed at expressing the topics that the user has consumed so far. We also tried using an exponential decay hyper-parameter that is meant to give lower weight to items seen farther away in the past, and thus better accommodate a non-stationary setting, however the best results were obtained with the simpler, and perhaps more stable, topic average.

In terms of the actual distance function d, we discovered that the Euclidean distance $d(\boldsymbol{\theta}_t, \boldsymbol{h}_{t-1}) = ||\boldsymbol{\theta}_t - \boldsymbol{h}_{t-1}||$ did poorly at identifying surprising items because it was often dominated by many topics that were in common between different items, even when they had small probabilities. Instead, to calculate surprise we use the topic that stands out the most with respect to the maximum topic probability so far:

$$d(\boldsymbol{\theta}_t, \boldsymbol{h}_{t-1}) = ||\boldsymbol{\theta}_t - \boldsymbol{m}_{t-1}||_\infty = \max_{1 \leq k \leq K}(\theta_{t,k} - m_{t-1,k})$$
$$\text{where} \quad m_{t-1,k} = \max_{1 \leq z \leq t-1} h_{z,k} \quad (15)$$

The vector $\boldsymbol{m}_{t-1} = [m_{t-1,k}]_{k=1..K}$ stores for each topic k the largest probability with which it was seen across all the books read so far by the user u.

The preference vector is computed in the basic model as the weighted average of the topic distributions seen by the user, using their ratings as weights:

$$\boldsymbol{p}_t = \frac{1}{t-1}\sum_{z=1}^{t-1} r_z \boldsymbol{\theta}_z \quad (16)$$

Here too we tried using an exponentially decaying weights for past items, however best performance was obtained using the simple weighted average.

5 TOPIC-LEVEL SURPRISE AND SERENDIPITY DATASET

We use as raw data source the Goodreads dataset [36], which consists of over 15M reviews for about 2M books from around 465K users. For lack of access to the actual book contents, we associate each book ID with an artificial *book content* that is created by concatenating all its English reviews and trimming to a maximum of 10K tokens. We extract a main set of 1,294,532 Books and their reviews by considering only books with at least 50 tokens. We create a smaller subset LDABooks for training an LDA topic model using only the 303,832 books that contained at least 1,000 tokens in their book content. Furthermore, we extracted a main set of 26,374 Users by considering only those who read and rated at least 100 different items from the Books set. While this may lead to selection bias, it was done to minimize the chance of including users with missing book ratings, based on the assumption that users who submit many ratings are more likely to report every book they read on Goodreads. Overall, the reading histories of these Users contain 1,043,437 unique books.

5.1 Manual Annotation of Topic-Level Surprise

To enable evaluation of the various serendipity models introduced above, we sampled a set of 4 *reference users* with reading histories that appeared to be diverse, in order to increase the likelihood of finding books that are topic-level surprising. For every book in a user's reading history, we read through the book content (concatenated reviews) in order to (a) manually create a *list of the major topics* for that book; and (b) use that list and the book content to create a *short summary* of the book. Sometimes the accumulated book content was too small to get a clear idea of the book topics,

Table 1: Reference users statistics. The total number of books for each user is 15 more than the number of manually labeled books.

	User 1	User 2	User 3	User 4	Total
Books read	140	114	137	118	509
Manually labeled books	125	99	122	103	449
Surprising books	10	13	14	17	54

as such we queried Goodreads and Amazon for additional reviews that were then added to the book content field.

For each of the 4 reference users, we manually annotate the books in their reading history with binary surprise labels. We skip the first 15 books, as these will be used as a burn-in set to learn a more stable estimate of the user's initial preference vector. Starting with position 16 in the time series of books, we compare its list of major topics and the book summary with a running summary of the topics of the books read so far, as well as their summaries. If the current book has topics that have not appeared in the previous books, or that were only tangentially addressed in previous books, then the book is labeled as topic-level surprising. Note that topic-level surprise is only one of many types of surprise, some of which are even independent of content, e.g. the user discovering a book by accident [12]. Table 1 shows annotation statistics for each reference user, including the number of surprising books. Overall, this is a time consuming, cognitively demanding annotation exercise. Although the total number of books that are manually labeled for surprise is not small, the requirement and difficulty of annotating each book with respect to the previously read books limited the annotation of book reading histories to only 4 users. To facilitate reproducibility and future progress in this area, we make the code and the dataset publicly available at https://github.com/Tonmoy/surprise-and-serendipity.

6 EXPERIMENTAL EVALUATION

We train a Latent Dirichlet Allocation (LDA) [5] topic model with $K = 100$ topics on the LDABooks set, and use it to generate the input topic distributions for all Books. We evaluate a total of 6 models: (1) Bayesian linear regression BLR(β, τ_s, τ_d, N); (2) variance bounded Bayesian linear regression vbBLR(β, τ_v, τ_s, τ_d, N); (3) AROW regression AROW(r_1, r_2, τ_s, τ_d, N); (4) normalized least mean square NLMS(η, τ_s, τ_d, k, N), (5) the hybrid combination AROW+vbBLR; and (6) the basic model Basic(τ_s, τ_d, N). The hyperparameters for each model are indicated between parentheses and are tuned using a *leave-one-out* setup: if reference user $u \in U$ is the current test user, then we select the hyperparameter values that lead to the best average F_1 on the other 3 users in $U - \{u\}$. This is repeated 4 times, in order to get test results on all reference users. For each method, user preference vectors $\boldsymbol{p}_t(u)$ are created for every user u in Users at every time step t in their time series of books. To compute precision and recall for the task of serendipity recommendation, we use the procedure shown in Algorithm 2 for each reference user. We use a similar procedure for evaluating only for surprise recommendation by removing the tests for positive ratings from Algorithms 1 and 2.

Algorithm 2: EvaluateSerendipityForUser(u)

Input: A reference user u; τ_s is the surprise threshold; $mSur(u, i+1)$ is true *iff* item $i+1$ was manually labeled as surprising for u.
Output: The Precision (P), Recall (R), and F1 measure (F_1) for reference user u.

$tp, tn, fp, fn \leftarrow 0$;
for *each item i in user u's time series* **do**
  if $\langle v, j \rangle$ = FindSerendipity(u, i) *is not null* **then**
    if $Sur(v, j+1) > \tau_s$ **then**
      if $mSur(u, i+1)$ and $r_{i+1} > 0$ **then** $tp \leftarrow tp + 1$
      else $fp \leftarrow fp + 1$
    else
      if $mSur(u, i+1)$ and $r_{i+1} > 0$ **then** $fn \leftarrow fn + 1$
      else $tn \leftarrow tn + 1$
$P = tp/(tp + fp)$, $R = tp/(tp + fn)$, $F_1 = 2PR/(P + R)$;
return P, R, F_1

The surprise recommendation results are listed in Table 2 and show the AROW model outperforming all other models, achieving an average F1 of 56.5%. The Basic model has the lowest average F1 of 42.7%. The results also show that models that utilize Bayesian surprise outperform other models by a considerable margin. The last two rows show the random baseline performance using either a surprise probability of 0.5, or the ratio of surprising books observed in the data. The serendipity recommendation performance is reported in Table 3. Because AROW obtained the best surprise recommendation performance and vbBLR the best serendipity recommendation, we also evaluated a hybrid AROW+vbBLR combination, where AROW is used for computing surprise, and vbBLR is used for identifying the most similar users in Algorithm 1. This combination outperforms all other models, obtaining an average F1 of 37.0%. The Basic model's performance is very unstable; while it obtains competitive performance on the first 3 users, on user 4 all the book items that it recommends as serendipitous are wrong, leading to no true positives and consequently zero F1. A possible reason is because the hyperparameters that obtain the best performance when tuned on the first 3 users are not suitable for user 4. However, even tuning on the user 4 itself lead to a very low F1 of around 13%.

Error analysis for the best models reveals that most errors are caused by (a) reviewers writing about topics that are unrelated to the book content, such as movies made based on the book, or books written by the same author; and (b) LDA creating irrelevant topics based on character names and other proper names.

7 BEYOND-ACCURACY RECOMMENDATION METRICS

McNee et al. [25] observed that focusing solely on the accurate ranking of items that are known or expected by users misses other important aspects that can further amplify user satisfaction. Currently, the recommender systems literature ([14, 18, 22], *inter alia*) recognizes five major non-accuracy aspects: serendipity, unexpectedness, novelty, diversity, and coverage. As discussed in Section 1, diversity refers to how dissimilar generated recommendations are,

Table 2: Surprising item recommendation performance (%) evaluated across 449 manually annotated books from 4 reference users.

	User 1			User 2			User 3			User 4			All
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	Avg F1
BLR	30.0	60.0	40.0	50.0	69.2	58.1	64.3	64.3	64.3	55.6	58.8	57.1	54.9
vbBLR	23.1	60.0	33.3	50.0	69.2	58.1	64.3	64.3	64.3	52.3	64.7	57.9	53.4
AROW	30.0	60.0	40.0	61.5	61.5	61.5	64.3	64.3	64.3	62.5	58.8	60.0	**56.5**
NLMS	30.0	60.0	40.0	34.6	69.2	46.2	50.0	35.7	41.7	33.3	64.7	44.0	43.0
Basic	20.0	30.0	24.0	44.4	61.5	51.6	66.7	42.9	52.2	54.5	35.3	42.9	42.7
Random ($p = 0.5$)	8.0	50.0	13.8	13.1	50.0	20.8	11.4	50.0	18.6	16.5	50.0	24.8	19.5
Random ($p = P/T$)	8.0	8.0	8.0	13.1	13.1	13.1	11.4	11.4	11.4	16.5	16.5	16.5	12.3

Table 3: Serendipitous item recommendation performance (%), across 449 manually annotated books from 4 reference users.

	User 1			User 2			User 3			User 4			All
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	Avg F1
BLR	13.8	57.1	22.2	30.8	80.0	44.4	22.7	41.7	29.4	22.7	50.0	31.3	31.8
vbBLR	16.7	71.4	27.0	30.0	90.0	45.0	20.0	41.7	27.0	30.0	60.0	40.0	**34.8**
AROW	14.3	57.1	22.9	31.3	50.0	38.5	23.8	41.7	30.3	22.7	50.0	31.3	30.8
NLMS	12.1	57.1	20.0	18.8	33.3	24.0	15.4	36.4	21.6	17.9	50.0	26.3	23.0
Basic	16.1	71.4	26.3	32.0	72.7	44.4	44.4	33.3	38.1	0.00	0.00	0.00	27.2
AROW+vbBLR	19.2	71.4	30.3	31.8	70.0	43.8	22.7	41.7	29.4	35.3	60.0	44.4	**37.0**
Random ($p = 0.5$)	5.6	50.0	10.1	11.1	50.0	18.2	9.8	50.0	16.4	10.7	50.0	17.6	15.6
Random ($p = P/T$)	5.6	5.6	5.6	11.1	11.1	11.1	9.8	9.8	9.8	10.7	10.7	10.7	9.3

whereas coverage reflects the degree to which they cover the entire spectrum of available items. The remaining three concepts, namely serendipity, unexpectedness, and novelty, have been variously described using multiple definitions that often result in substantial overlap, subsumption, and sometimes even identity, between their conceptual domains. For example, noting that there is no consensus on the definition of serendipity, Kotkov et al. [19] investigate eight definitions, starting from a common view where serendipity has three components – relevance, novelty and unexpectedness, each of which has multiple variations. Novelty typically refers to a user being unfamiliar with a recommended item [28], a desirable property that is lacking when recommendation lists contain only items that are popular or well-known [16]. At the same time, Oh et al. [28] note that sometimes novel recommendations are equated with diversified recommendations, whereas other approaches define novel items more broadly as any item that widens a user's interests. Definitional ambiguity aside, other than serendipity, unexpectedness is most related to our notion of surprise. Li and Tuzhilin [22] define the unexpectedness of an item as the distance between that item and the closure of all previously consumed items, computed in a latent embedding space, which is conceptually similar to how surprise is defined in the basic approach from Section 4. A hybrid utility function is then defined as a linear combination of the item's predicted rating and its unexpectedness. In contrast, we define serendipity as a multiplicative combination of the post-factum, actual item rating with its estimated Bayesian surprise, and use collaborative filtering to identify items with high potential for serendipity.

8 CONCLUSION AND FUTURE WORK

We introduced a method for recommending serendipitous items where serendipity is defined in terms of Bayesian surprise. To facilitate its computation, we proposed adaptations of online learning algorithms for the non-stationary setting of user preferences. Experiments show that methods rooted in Bayesian surprise obtain superior results. Future work includes an expanded dataset, non-linear models, deep topic modeling, and surprise measures that go beyond topic-level. Since Bayesian linear regression can be obtained as a special case of the Kalman filter, an intriguing future direction is adapting a Kalman filter for continual online learning [34] of the non-stationary user preferences.

ACKNOWLEDGMENTS

We would like to thank Cade Mack for initial implementations of the BLR and vbBLR models and their evaluations on artificial data. We are also grateful to the anonymous reviewers for their constructive comments. This research was partly supported by the United States Air Force (USAF) under Contract No. FA8750-21-C-0075. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the USAF.

REFERENCES

[1] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On Over-Specialization and Concentration Bias of Recommendations: Probabilistic Neighborhood Selection in Collaborative Filtering Systems. In *Proceedings of the 8th ACM Conference on Recommender Systems* (Foster City, Silicon Valley, California, USA) *(RecSys '14)*. Association for Computing Machinery, New York, NY, USA, 153–160. https://doi.org/10.1145/2645710.2645752

[2] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On Unexpectedness in Recommender Systems: Or How to Better Expect the Unexpected. *ACM Trans. Intell. Syst. Technol.* 5, 4, Article 54 (dec 2014). https://doi.org/10.1145/2559952

[3] Neil Bershad. 1986. Analysis of the normalized LMS algorithm with Gaussian inputs. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 34, 4 (1986), 793–806.

[4] C.M. Bishop. 2013. *Pattern Recognition and Machine Learning: All "just the Facts 101" Material.* Springer.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3, null (mar 2003), 993–1022.

[6] Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Melbourne, Australia) *(SIGIR '98)*. Association for Computing Machinery, New York, NY, USA, 335–336. https://doi.org/10.1145/290941.291025

[7] Li Chen, Yonghua Yang, Ningxia Wang, Keping Yang, and Quan Yuan. 2019. How Serendipity Improves User Satisfaction with Recommendations? A Large-Scale User Evaluation. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 240–250. https://doi.org/10.1145/3308558.3313469

[8] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. 2017. Learning to Recommend Accurate and Diverse Items. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 183–192. https://doi.org/10.1145/3038912.3052585

[9] Koby Crammer, Alex Kulesza, and Mark Dredze. 2009. Adaptive Regularization of Weight Vectors. In *Advances in Neural Information Processing Systems*, Vol. 22. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2009/hash/8ebda540cbcc4d7336496819a46a1b68-Abstract.html

[10] Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine learning*. Cambridge University Press, 264–271.

[11] Seth Flaxman, Sharad Goel, and Justin M. Rao. 2016. Filter Bubbles, Echo Chambers, and Online News Consumption. *Public Opinion Quarterly* 80, S1 (March 2016), 298–320. https://doi.org/10.1093/poq/nfw006 _eprint: https://academic.oup.com/poq/article-pdf/80/S1/298/17120810/nfw006.pdf.

[12] Zhe Fu, Xi Niu, and Li Yu. 2023. Wisdom of Crowds and Fine-Grained Learning for Serendipity Recommendations. In *Proceedings of The 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

[13] Zhaolin Gao, Tianshu Shen, Zheda Mai, Mohamed Reda Bouadjenek, Isaac Waller, Ashton Anderson, Ron Bodkin, and Scott Sanner. 2022. Mitigating the Filter Bubble While Maintaining Relevance: Targeted Diversification with VAE-based Recommender Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2524–2531. https://doi.org/10.1145/3477495.3531890

[14] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems* (Barcelona, Spain) *(RecSys '10)*. Association for Computing Machinery, New York, NY, USA, 257–260. https://doi.org/10.1145/1864708.1864761

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[16] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. Inf. Syst.* 22, 1 (jan 2004), 5–53. https://doi.org/10.1145/963770.963772

[17] Laurent Itti and Pierre Baldi. 2005. Bayesian Surprise Attracts Human Attention. In *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt (Eds.), Vol. 18. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2005/file/0172d289da48c48de8c5ebf3de9f7ee1-Paper.pdf

[18] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 2 (dec 2016), 42 pages. https://doi.org/10.1145/2926720

[19] Denis Kotkov, Joseph A. Konstan, Qian Zhao, and Jari Veijalainen. 2018. Investigating Serendipity in Recommender Systems Based on Real User Feedback. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* (Pau, France) *(SAC '18)*. Association for Computing Machinery, New York, NY, USA, 1341–1350. https://doi.org/10.1145/3167132.3167276

[20] Solomon Kullback. 1959. *Information Theory and Statistics.* Wiley.

[21] Pan Li, Maofei Que, Zhichao Jiang, YAO HU, and Alexander Tuzhilin. 2020. PURS: Personalized Unexpected Recommender System for Improving User Satisfaction. In *Proceedings of the 14th ACM Conference on Recommender Systems* (Virtual Event, Brazil) *(RecSys '20)*. Association for Computing Machinery, New York, NY, USA, 279–288. https://doi.org/10.1145/3383313.3412238

[22] Pan Li and Alexander Tuzhilin. 2020. Latent Unexpected Recommendations. *ACM Trans. Intell. Syst. Technol.* 11, 6, Article 70 (sep 2020), 25 pages. https://doi.org/10.1145/3404855

[23] Xueqi Li, Wenjun Jiang, Weiguang Chen, Jie Wu, Guojun Wang, and Kenli Li. 2020. Directional and Explainable Serendipity Recommendation. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 122–132. https://doi.org/10.1145/3366423.3380100

[24] Masoud Mansoury, Himan Abdollahpouri, Mykola Pechenizkiy, Bamshad Mobasher, and Robin Burke. 2020. Feedback Loop and Bias Amplification in Recommender Systems. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 2145–2148. https://doi.org/10.1145/3340531.3412152

[25] Sean M. McNee, John Riedl, and Joseph A. Konstan. 2006. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems (CHI EA '06)*. Association for Computing Machinery, New York, NY, USA, 1097–1101. https://doi.org/10.1145/1125451.1125659

[26] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. 2002. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Eighteenth National Conference on Artificial Intelligence* (Edmonton, Alberta, Canada). American Association for Artificial Intelligence, USA, 187–192.

[27] Xi Niu, Fakhri Abbas, Mary Lou Maher, and Kazjon Grace. 2018. Surprise Me If You Can: Serendipity in Health Information. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173597

[28] Jinoh Oh, Sun Park, Hwanjo Yu, Min Song, and Seung-Taek Park. 2011. Novel Recommendation Based on Personal Popularity Tendency. In *2011 IEEE 11th International Conference on Data Mining*. 507–516. https://doi.org/10.1109/ICDM.2011.110 ISSN: 2374-8486.

[29] Gaurav Pandey, Denis Kotkov, and Alexander Semenov. 2018. Recommending Serendipitous Items Using Transfer Learning. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (Torino, Italy) *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 1771–1774. https://doi.org/10.1145/3269206.3269268

[30] E. Pariser. 2011. *The Filter Bubble: What The Internet Is Hiding From You.* Penguin Books Limited, London, UK.

[31] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. 2012. A literature review and classification of recommender systems research. *Expert Systems with Applications* 39, 11 (Sept. 2012), 10059–10072. https://doi.org/10.1016/j.eswa.2012.02.038

[32] Paul Resnick and Hal R. Varian. 1997. Recommender Systems. *Commun. ACM* 40, 3 (mar 1997), 56–58. https://doi.org/10.1145/245108.245121

[33] Rahul Shrivastava, Dilip Singh Sisodia, Naresh Kumar Nagwani, and Upendra Roy BP. 2022. An optimized recommendation framework exploiting textual review based opinion mining for generating pleasantly surprising, novel yet relevant recommendations. *Pattern Recognition Letters* 159 (2022), 91–99.

[34] Michalis K. Titsias, Alexandre Galashov, Amal Rannen-Triki, Razvan Pascanu, Yee Whye Teh, and Jorg Bornschein. 2023. Kalman Filter for Online Classification of Non-Stationary Data. arXiv:2306.08448 [cs.LG]

[35] Nina Vaits and Koby Crammer. 2011. Re-adapting the regularization of weights for non-stationary regression. In *Proceedings of the 22nd international conference on Algorithmic learning theory (ALT'11)*. Springer-Verlag, Berlin, Heidelberg, 114–128.

[36] Mengting Wan and Julian McAuley. 2018. Item Recommendation on Monotonic Behavior Chains. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 86–94. https://doi.org/10.1145/3240323.3240369

[37] Wenjie Wang, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2022. User-controllable Recommendation Against Filter Bubbles. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 1251–1261. https://doi.org/10.1145/3477495.3532075

[38] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1, Article 5 (feb 2019), 38 pages. https://doi.org/10.1145/3285029

[39] Yu Zheng, Chen Gao, Liang Chen, Depeng Jin, and Yong Li. 2021. DGCN: Diversified Recommendation with Graph Convolutional Networks. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 401–412. https://doi.org/10.1145/3442381.3449835

[40] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web (WWW '05)*. Association for Computing Machinery, New York, NY, USA, 22–32. https://doi.org/10.1145/1060745.1060754