Probabilistic Topic Modeling, Reinforcement Learning, and Crowdsourcing for Personalized Recommendations

Evangelos Tripolitakis and Georgios Chalkiadakis

School of Electrical and Computer Engineering, Technical University of Crete, Greece,
vtripolitakis@intelligence.tuc.gr, gehalk@intelligence.tuc.gr,
WWW home page: http://www.ece.tuc.gr/

Abstract. We put forward an innovative use of probabilistic topic modeling (PTM) intertwined with reinforcement learning (RL), to provide personalized recommendations. Specifically, we model items under recommendation as *mixtures of latent topics* following a distribution with Dirichlet priors; this can be achieved via the exploitation of *crowdsourced information* for each item. Similarly, we model the *user herself* as an "evolving" document represented by its respective mixture of latent topics. The user's topic distribution is appropriately updated each time she consumes an item. Recommendations are subsequently based on the divergence between the topic distributions of the user and available items. However, to tackle the exploration versus exploitation dilemma, we apply RL to vary the user's topic distribution update rate. Our method is immune to the notorious "cold start" problem, and it can effectively cope with changing user preferences. Moreover, it is shown to be competitive against state-of-the-art algorithms, outperforming them in terms of sequential performance.

Keywords: Recommender Systems, Applications of Reinforcement Learning, Graphical Models, Crowdsourcing

1 Introduction

In this paper, we introduce a system that provides high quality, personalized recommendations, with minimal user interaction and without relying to others' ratings. We achieve that by *(a)* employing the key assumption that *users can be viewed as the amalgamation of items they consume* and *(b)* by exploiting, to the best extent possible, available *crowdsourced information* related to the items.

Our solution takes into account the *richness*, *complexity*, and *evolution* of user preferences by exploiting *latent* features, while at the same time identifying preference shifts, and adapting in a dynamic, transparent manner. Regarding the *cold-start* problem, we are able to perform high-quality suggestions. Further, the system presents computational efficiency, thus suitable to cope with large-scale data sets. Last, it designed to be domain-agnostic, hence minimizing the fine-tuning effort from the practitioner. Our work builds on the *"you are what you consume"* approach put forward by [3], but, crucially, employs *probabilistic topic modeling (PTM)* [18] and intertwines it with *reinforcement learning (RL)* [9] techniques.

Specifically, we model *both items* and *users* as distributions of topics with *Dirichlet priors* over *crowdsourced corpora* of documents, found in on-line communities producing peer-reviewed, unbiased information—here we use Wikipedia movie synopses. We thus replace *explicit* attributes (e.g. movie categories) by *latent* features in *both* user and item distributions. We adapt our recommendations policy using concepts from the RL literature. We inflict drastic changes to the user model *only* if there are strong indications of a decaying predictions performance. This is accomplished by the alteration of the system's *learning rate*: minimum changes are imposed if performance is strong when compared to its average within a past ratings window; on the other hand, we increase the learning rate significantly in order to move out of a series of low-rated recommendations. Moreover, we adopt an ϵ-greedy exploration strategy that occasionally selects random items to avoid "local maxima" in the recommendations space. Last but not least, we successfully tackle user preference shifts.

In summary, our approach *(i)* requires no clustering, does not predict ratings, and does not use hard-coded attributes (e.g. categories, genres, etc.); *(ii)* exploits crowdsourced information, which is widely available—and usually implies good quality, objective, peer-reviewed sources of information; *(iii)* requires minimal user interaction limited to user ratings;*(iv)* is unbiased and immune to external noise and "information poisoning", as it does not consider other users' behaviors: it just adheres to user's preferences; *(v)* it can cope with the "cold-start" problem (i.e. given limited or no background information about new users, we should be able to adjust the user interaction with the system in order to suggest proper items after a small number of iterations) since it does not require other users' ratings; *(vi)* it can cope with evolving user preferences, and even mood-changes; on the technical side, intertwining PTMs with RL methods is innovative and interesting; *(vii)* it provides intuition on the actual user interests, since these correspond to the PTM-inferred topics that relate to the particular user mostly (i.e., the most frequent ones in its associated topics distribution); *(viii)* allows for the easy online update and enrichment of items and user models and *(ix)* can scale-up by employing industry-proven algorithms; *(x)* it is not domain-specific, and could be used for cross-domain recommendations.

2 Related work

Collaborative filtering (CF) techniques exploit ratings from similar users or items and try to predict the user's preference for each item, ignoring other background information [15, 19, 2, 17]. CF algorithms received acclaim on the Netflix contest [10–12]. Nevertheless, CF suffers from the *cold-start* problem, requiring the a certain amount of ratings to operate. Similarly, newly-introduced items need a minimum number of ratings. The same issue applies to CF variations, where a considerable amount of transactions (e.g. purchases) needs to be available.

Content-based recommender systems have been proposed to work along with CF algorithms and form hybrid systems. Melville et al. [5] proposed a content-based predictor to boost existing user data and exploit it for personalized recommendations using CF. Hoffman [16] introduced latent class variables in a mixture model setting, to search for user communities and profiles that present special interest.

[2] have also proposed the use of PTMs [20, 1] for making recommendations for scientific articles; while [21] used topic modeling (via a generative model based on Latent Dirichlet Allocation) to exploit movie reviews and boost the performance of a CF system. These works, however, still rely on the ratings of multiple users (in the spirit of CF), and do not maintain an evolving user model, as we do in this paper—nor do they intertwine PTM with RL techniques.

We build on the work of [3] who, however represent users and items by *multivariate Gaussian distributions* over movie categories; our use of PTMs is a generic method providing immediate insights about user preferences, via the topics describing the user.

3 Theoretical Background

Here, we provide some background on PTMs and the key RL techniques used in this work. PTMs treat data as arising from a generative process that defines a joint probability distribution over observed and hidden variables corresponding to the underlying topic structure of a document, and then apply tractable Bayesian inference (e.g., sampling-based) methods to estimate the posterior distributions of the hidden variables.

In the *Latent Dirichlet allocation (LDA)* [1], all M documents are represented by mixtures of K topics with Dirichlet priors and N words in total. The parameters are the following: $\boldsymbol{\alpha}$ is the parameter (a vector of positive reals) of the Dirichlet prior on the per-document topic distributions, $\boldsymbol{\beta}$ is the parameter (a vector of positive reals) of the Dirichlet prior on the per-topic word distribution, θ_i is the topic distribution for document i, ϕ_k is the word distribution for topic k, z_{ij} is the topic for the jth word in document i, and w_{ij} is the specific word. The LDA employs the following processes:

1. Select $\theta_i \sim Dir(\boldsymbol{\alpha})$, $Dir(\boldsymbol{\alpha})$: the Dirichlet distribution for $\boldsymbol{\alpha}$, for the document $i \in \{1..M\}$
2. Select $\phi_k \sim Dir(\boldsymbol{\beta})$, $Dir(\boldsymbol{\beta})$: the Dirichlet distribution for $\boldsymbol{\beta}$, for the word $k \in \{1..N\}$
3. For each word $w_{ij} : i \in \{1..M\}, j \in \{1..N_i\}$, with N_i being the number of words of document i
   (a) Select $z_{ij} \sim Multinomial(\theta_i)$
   (b) Select $w_{ij} \sim Multinomial(\phi_{z_{ij}})$

In the MAS and RL literature several algorithms have been suggested for the variation of a *learning rate*, which governs the extent to which an agent's decisions are influenced by recent knowledge [9]. We focus on the WoLF ("Win or Learn Fast") principle adopting the concept of taking drastic measures when the agent is losing, and make small or no changes in case of a win streak [7, 8]. The decision is made considering the relation between the overall average performance and the performance within the k last steps. Hence, we introduce a δ_{win} and a δ_{lose} variation to the learning rate, for the cases the recommender predictions keep performing well, or worsen, respectively. Further, we adopt an ϵ-greedy strategy, to combat the "exploration vs. exploitation" dilemma [9], the recommender will suggest the most preferable item with probability $1 - \epsilon$; and chooses a random item otherwise.

4 Our model

We offer personalized recommendations to a set of Z users $U = \{u_1, ..., u_Z\}$, taken from a set of *M* available items $Y = \{y_1, ..., y_M\}$. Users are first asked to select from a set at least one movie they like most. Upon this, the system returns a suggestion and asks users to rate it, in order to update their model. We use LDA to model both items and users as mixtures over topics with a *Dirichlet* prior. A set of assumptions are made for this model to apply: a) There is descriptive information freely available for all items, b) The descriptions are non-trivial. This means that we need at least a couple of paragraphs of text, c) The descriptions are objective and do not contain other information than that which illustrates the characteristics / behavior / function of the item.

4.1 Item modeling

Items $y_i \in Y, |Y| = M, 1 \leq i \leq M$ are represented by *documents* belonging to a corpus D, containing M documents. There are N words in total in the corpus *vocabulary*. We set the number of *topics* associated with the corpus to K. For each document, we choose:

1. $\theta_i \sim Dir_K(\boldsymbol{\alpha})$, where θ_i is the distribution of topics in document i, and $Dir_K(\boldsymbol{\alpha})$ is the Dirichlet distribution of parameter $\boldsymbol{\alpha}$.
2. $\phi_k \sim Dir_N(\boldsymbol{\beta})$, where ϕ_k is the word distribution for topic k, and $Dir_N(\boldsymbol{\beta})$ is the Dirichlet distribution of parameter $\boldsymbol{\beta}$

$\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the Dirichlet parameters of topic distributions per document and word distributions per topic, respectively.

4.2 User modeling

Similarly, each user u_j is modeled after a document represented by an evolving mixture of topics with Dirichlet prior. The distribution of topics mixture, follows, like items, a Dirichlet distribution. Hence: $\theta_j \sim Dir_K(\boldsymbol{\alpha})$, where θ_j is the distribution of topics in the single document that models the user, $Dir_K(\boldsymbol{\alpha})$ is the Dirichlet distribution of parameter $\boldsymbol{\alpha}$, and K the number of topics.

4.3 Item-model and user-model updating

After the necessary preprocessing steps (i.e. stop word removal, stemming, and so on), the input corpus consisting of M crowdsourced documents is used to derive the final LDA model. This procedure is repeated for various number of topics K, until we reach a number that yields the lowest *perplexity* [22] score in a test set consisting of M_{test} documents. A simplified, tractable approach is viable via approximation methods utilizing sampling by [23]:

$$perplexity(D_{test}) = exp\{-\frac{\sum_{d=1}^{M_{test}} logp(\mathbf{w}_d)}{\sum_{d=1}^{M_{test}} N_d}\} \quad (1)$$

The number of words of document d belonging to the test set, is N_d. A decrease in perplexity is an indication of an increase in the predictive strength of the model.

Apart from the initial modeling of a set of available documents, new documents still need to be added to the corpus D. There are two alternatives: *(a)* model recalculation in batches; and *(b)* on-line LDA which is described by [24] and allows on-the-fly incorporation of new documents on the model. In our case, the corpus is fixed. In real-world environments, depending on the size and complexity of available documents, the practitioner is free to choose a suitable solution.

Then, in order to update the user-model, we need to incorporate three key factors: *(a)* The user's ratings of items that she consumes; *(b)* the possibility that she favors other types of items than those she used to, hence reflecting a mood shift; and *(c)* a potential exhaustion of preferable items.

Each time the user consumes an item, she provides a rating. This rating is used in the Bayesian updating of her topic mixture. As the topic distribution of the document that models the user is unknown, we utilize Bayes rule to take into account *observations* (user ratings), a *likelihood function* and a *marginal probability*, so as to derive a *posterior* distribution.

The likelihood function associates the prior with the observations, while preserving the form of the overall model. The posterior which is produced represents the updated belief for the prior, given the evidence. Using the posterior beliefs $f(\boldsymbol{\theta}|y)$, we are able to update our unknown model. In our case we use the *Dirichlet* and the *multinomial* distributions, which are conjugate. This is a useful property, which allows us to perform easy updates to the prior's *hyperparameters*, using a closed form equation, we will show below.

Given documents $\mathbf{y}$ and with topics mixtures $\boldsymbol{\theta}$, we have:

$$\boldsymbol{\theta} \sim Dir(\boldsymbol{\alpha} = \langle a_1, ..., a_K \rangle) \tag{2}$$

$$\mathbf{y} \sim Mult(\boldsymbol{\theta} = \langle \theta_1, ..., \theta_K \rangle) \tag{3}$$

In detail, the topic mixtures $\boldsymbol{\theta}$ are described by:

$$Dir(\boldsymbol{\theta}|a_1, ..., a_K) = \frac{\Gamma(a_1 + ... + a_K)}{\Gamma(a_1)...\Gamma(a_K)} \prod_{i=1}^{K} \theta_i^{a_i - 1} \tag{4}$$

Given the evidence (document y) the posterior becomes :

$$\begin{aligned} f(\boldsymbol{\theta}|y) \propto f(\boldsymbol{\theta}, y) &= f(\theta_1, ..., \theta_K|\alpha_1, ..., \alpha_K) f(y|\theta_1, ..., \theta_K) \\ &\propto \prod_{j=1}^{K} \theta_j^{a_j - 1} \prod_{j=1}^{K} \theta_j^{y^{(j)}} = \prod_{j=1}^{K} \theta_j^{a_j - 1 + (y^{(j)})} \end{aligned} \tag{5}$$

The updated hyperparameters of the Dirichlet prior are:

$$a_j{}' = a_j + y^{(j)} \tag{6}$$

Therefore, we can update the user model by simply adding to each topic θ_k the respective topic counts from the document y that was just rated. Furthermore, we incorporate the user rating by repeating this step n times. For a user, depending on the t-th rating r of an item that was recommended, we update the user model by taking n samples, using:

$$n \propto {\Delta_t}^r - 1 \tag{7}$$

where: Δ_t is the variable learning rate (a real number) for the t-th recommended item and r is the actual rating.

Items that have been positively rated by the user, can be thought as having greater *influence* on the overall user preferences. Further, the intuition behind the Eq. 7 is that items rated by 0,1 and 2 should have minimal or no influence on the evolution of the user's model. Contrary to that, items rated between 3 and 5 should contribute proportionally to their significance. We empirically found suitable values of Δ_t to lie between 1.0 and 4.0. Adjusting this variable learning rate allows us to move away from less promising areas of the search space (see *Alg. 1*).

4.4 Recommendation phase

The recommendation phase consists of two main functions: *a)* the querying of the available low-dimensional representation of items to obtain the most appropriate, and *b)* the monitoring of the system performance and adjustment of the learning rate. Once LDA has been run on the corpus and the initial user models have been built, the recommendation algorithm is displayed in the pseudocode of *Alg. 1*.

Given the fact that both users and items are represented by the same distribution, we assess their similarity by employing the cosine distance D_{cosine} metric, which is a common measure in the information retrieval domain:

$$D_{cosine}(P, Q) = 1 - \frac{\sum_{i=1}^{n} P_i \times Q_i}{\sqrt{\sum_{i=1}^{n} P_i} \times \sqrt{\sum_{i=1}^{n} Q_i}} \tag{8}$$

where P, Q are distributions of the same type and same size.

5 Experiments

We chose the *cinema movies* domain due its significance and popularity both scientifically-wise and business-wise. Instead of using domain-specific sites (e.g., IMDB), we chose Wikipedia as our source, which offers a large collection of lemmas with detailed plot summaries, actively checked for objectivity and clarity by a large number of contributors.

5.1 Information collection, pre-processing and topic model selection

We developed a workflow, which operates on *any on-line source* with an API, for initial information collection. Further, we employed the MALLET [25] topic modeling toolkit

Algorithm 1 Item recommendation

procedure RECOMMENDITEM
  with probability ϵ choose a random item, request a rating and go to **updateUser:**, with probability $1 - \epsilon$ continue
  for each item $y_i \in D$ **do**
    Calculate the cosine distance $D_{cosine}(y_i||u_j)$ between the item and the user u_j and store it in an array
  end for
  Find the (non-recommended) item with the smallest cosine distance, ask user for a rating
  updateUser:
  Update the average of user ratings $\overline{r_{u_j}}$ and the average rating for the latest ξ recommendations $\overline{r_{u_{j\xi}}}$
  if $\overline{r_{u_j}} > c\,\overline{r_{u_{j\xi}}}$ **then** % with $c \geq 1$
    $\Delta_{t+1} = \Delta_t + \delta_{lose}$
  else
    $\Delta_{t+1} = \Delta_t - \delta_{win}$
  end if
  Update user-model
end procedure

which implements the LDA algorithm (plus necessary pre-processing tools - e.g., stop words removal). Regarding topic number optimization, we ran LDA on two sets of documents from Movielens 1M and 10M datasets, consisting of 3,137 and 8,721 text documents (containing movie synopses) respectively. We split our corpus, using a ratio of 80%-20%, to yield training and test sets and calculated the perplexity for varying topic numbers according to Eq. 1. We found that the optimal number of topics is 280 and 240 respectively. We relaxed this to 75 topics as a trade-off between perplexity and computational cost.

Experimental setup Our algorithmic variants include: a) one with a fixed learning rate Δ, set to 4.0 and no exploration and b) one with variable $\Delta \in [1.0...4.0]$ and ϵ-greedy exploration set to 0.30. For both variants we update the user type taking n samples as in Eq. 7. Moreover, we optionally introduce "time-compensation" by multiplying (or not), the number of samples returned by Eq. 7 with an integer proportional (by a factor of 1.0 at our setup) to the current recommendation step number, hence favoring most recent ratings. We set $c = 1.1, \xi = 3, d_{win} = 0.05, d_{lose} = 0.3$. Last but not least, we validated the operation of our algorithm against a random setting, and investigated its adaptability to user mood shifts. For our experiments, we used 5 sets of 100 randomly-selected users from the Movielens 1M and 10M datasets[1], having 200 or more ratings each. We intentionally chose users with 200 or more ratings because we wanted to illustrate the non-trivial long-term learning behavior of our algorithm.

[1] Datasets: *Movielens 1M*: *1M ratings*, *6,040* users, *3,952* movies. *Movielens 10M*: *10M ratings*, *71,567* users, *10,681* and movies. We found nontrivial data on Wikipedia for *3,137* movies on the 1M dataset and *8,721* movies on the 10M dataset.

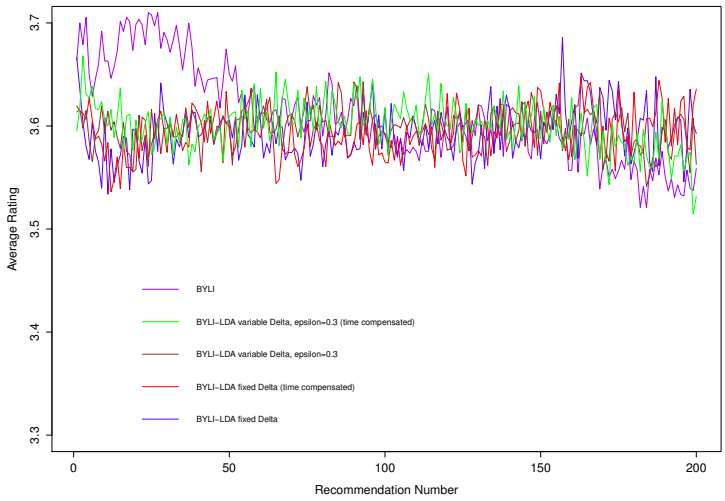

Fig. 1. BYLI-LDA vs BYLI; sequential recommendation performance on the 1M Movielens dataset

We compare our algorithm, *BYLI-LDA* with 2 existing methods: the "Bayes As You Like It (BYLI)", suggested Babas et al. (on the 1M dataset), and a sophisticated (with automatic retraining) implementation of the alternating least squares (ALS) algorithm for large, sparse matrix factorization (LSMF) by [17], on the Myrrix software. In detail, we check against a trained version of LSMF (using all database ratings for training, minus those of the 100 users in the set) and an untrained version of LSMF (as in [3]). Note that LSMF, being a CF method, has to first collect a small number of ratings in order to be able to return good recommendations, and thus cannot provide a meaningful recommendation at the first iteration.

The performance of recommendation algorithms is typically assessed by their Root Mean Square Error (RMSE); and RMSE calculation requires ratings predictions. Our algorithm, on the other hand, *does not* predict ratings, but simply suggests movies sequentially, based on its beliefs regarding the evolving user type. Actually, many algorithms do not explicitly calculate ratings, but just predict movie rankings. However, there is no standard way in the literature to transform a set of ranked results into predicted ratings, but these are typically calculated in a different way per algorithm (a fact that is justified by the suggestion of extra parameters such as user bias, item bias etc.[19]).

Regardless, we calculate RMSE by introducing a transformation function calculating the predicted rating of a recommendation: $r(i,j)_{predicted} \propto g(i,j)$. The transformation function $g(i,j)$ returns the predicted rating for an item i and a user j. We take a training set consisting of 80% of the movies each user has rated, and apply our algorithm. Next, we calculate the similarity vector of the movies in the training set and the user model, using our preferred similarity metric. Further, we calculate the distri-

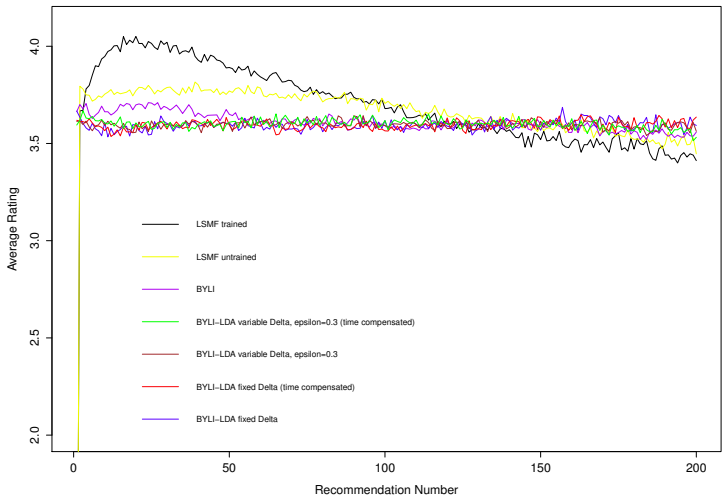

Fig. 2. BYLI-LDA vs BYLI and LSMF; sequential recommendation performance on the 1M Movielens dataset

bution of ratings of each user and divide the similarity vector into parts whose lengths correspond to the frequency of each rating. For the test set (20% of users' ratings) we apply again our algorithm, get the *user-item distance* and calculate the predicted rating $r(i,j)_{predicted}$ and calculate the RMSE. The average RMSE is 1.14542, higher compared to errors reported on the literature for this dataset, typically from 0.8 to 1.0. Since we do not predict ratings, however, this result is not a cause of major concern: As we detail below, our algorithm performs very well when evaluated *wrt.*: *(a)* sequential performance, indicated by average per recommendation ratings across 200 recommendations over 10 runs; and *(b)* average ratings across all recommendations. All averages were taken across the 5 user sets.

Results and evaluation on Movielens 1M Our algorithm presents a stable behavior in all its variants, as it is clear from Fig. 1. This is also clear from the standard deviation that is displayed on Table 1 ranging from 0.0966 to 0.1019. Clearly, the results from the variants with exploration perform slightly better compared to those without exploration. Furthermore, at closer inspection of Fig. 1 as the number of recommendations per user increases, all variants and especially those with exploration improve their recommendations.

Furthermore, as we see on Table 2, our algorithm performs similarly with BYLI, which is 0.18% better[2]. The variations that favor recent ratings perform slightly better than those who do not. Moreover, from the 60th recommendation onward, BYLI presents a decaying behavior, in contrast to BYLI-LDA whose sequential performance

[2] BYLI had been evaluated on MovieLens 1M only.

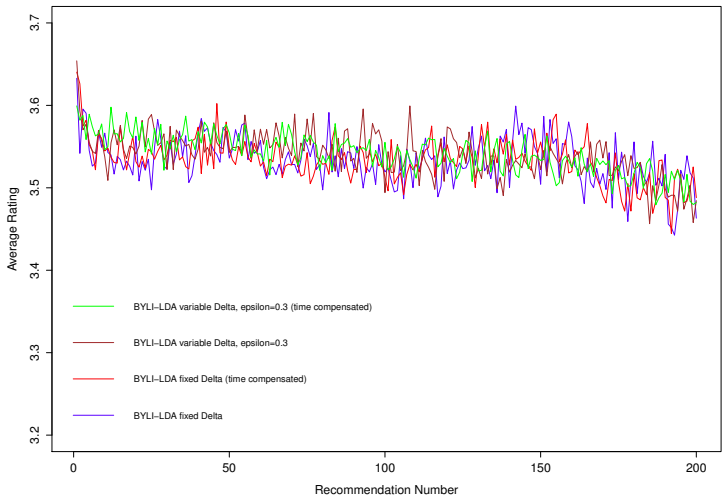

Fig. 3. BYLI-LDA Variants; performance on the 10M Movielens dataset

Table 1. Mean standard deviation

Variant	**Std**
Fixed Δ	0.09657605
Fixed Δ (time compensated)	0.09774386
Variable Δ/ϵ-greedy ϵ=0.3	0.1019049
Variable Δ/ϵ-greedy ϵ=0.3 (time compensated)	0.1009336

is far more stable. Regarding the LSMF, both its versions ("trained" and "untrained") present a decaying performance and after the 120th recommendation are outperformed by BYLI-LDA.

Results and evaluation on Movielens 10M In this dataset, the average rating is 3.512422 instead of 3.581564 at the 1M dataset. Similar to the 1M experiments, our algorithm displays a stable behavior here too, as Fig. 3 shows.

The recommendations are better than the dataset average on all variations of our algorithm. Contrary to that, as we see on Fig. 4, the trained version of LSMF performs much better compared to the 1M dataset, due to increase of the training dataset by 10 times. On the other hand, it still exposes a decaying behavior starting around the 55th recommendation and converges to our algorithm. As in the case of the 1M dataset, the variants with variable learning rate exploration perform better compared to those with fixed learning rate and no exploration. Moreover, the favoring of newer ratings over older ones produces slightly better results on the variants with variable learning rate

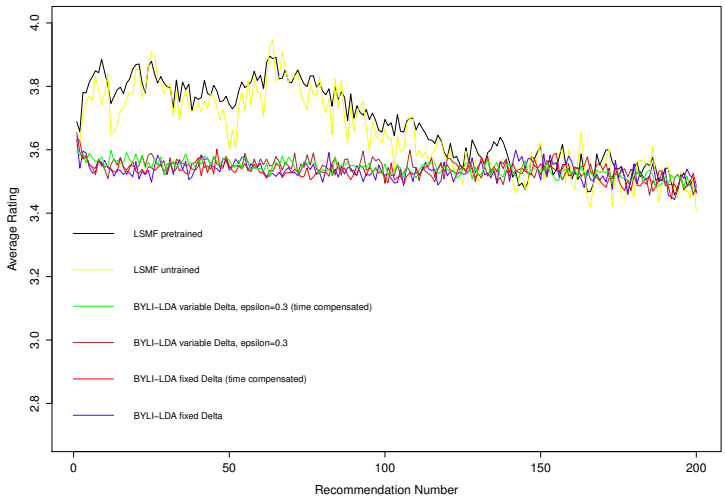

Fig. 4. BYLI-LDA vs LSMF; performance on the 10M Movielens dataset

and exploration. The opposite happens (as in the case of 1M) to the fixed learning rate variants. The results are presented on Table 2 as well.

5.2 Behavior against random

The MovieLens dataset consists of a large number of high ratings (yielding an average of 3.5815 out of 5). Hence, we expect random recommendations to offer adequate results, in terms of average score. Thus, to validate the proper operation of our method, it is essential to display its behavior against a random algorithm.

Our setup consists of 5 50-user sets. Users were exposed to 200 randomly chosen movies belonging to two distinct movie types, a) a *desirable* type rated by 5 and b) a *hated* type rated by 1. We see in Fig. 5 that BYLI-LDA without exploration recommends the best item until exhaustion of available *desirable* items, and switches to the recommendation of the *hated* items left. It is interesting to note the behavior of the ϵ-greedy variants of BYLI-LDA when using small values for ϵ (i.e., 0.02), Then, the inevitable repeated introduction of *hated* items, alters the user-type in order to encompass latent characteristics that allow the future recommendation of such items. Subsequently, the average rating exposes a decaying behavior which in addition, follows the mode switch around the 100th recommendation. Nevertheless, the average rating is distinct and significantly higher compared to that of the random setting, hence confirming the proper operation of our algorithm. By contrast, higher values of ϵ (0.2) introduce significant noise (in terms of a large volume of *hated* items), rapidly deteriorating the performance of the algorithm. Obviously, in a real setting, this does not occur, as the distribution of *hated* items is totally different. On the contrary, as our Movielens ex-

Table 2. Average ratings of all methods for Movielens 1M and Movielens 10M.

Methods	**1M**	**10M**
LSMF - pretrained	3.6848	3.677518
LSMF - untrained	3.6540	3.646337
BayesYouLikeIt (BYLI)	3.6112	-
BYLI-LDA - Variable Δ/ϵ-greedy ϵ=0.3 (time compensated)	**3.600598**	**3.541764**
BYLI-LDA - Variable Δ/ϵ-greedy ϵ=0.3	**3.59699**	**3.540057**
BYLI-LDA - Fixed Δ (time compensated)	**3.59421**	**3.533575**
BYLI-LDA - Fixed Δ	**3.593851**	**3.53371**
Dataset average	3.581564	3.512422

periments showed, exploration is absolutely desirable as it enhances the overall performance by avoiding local maxima.

Capturing changing user preferences We simulated the occurrence of a preference (or a "mood shift") on a user with 200 ratings equally split between a *desirable* item type: b and a *hated* type: a. The behavior of BYLI-LDA (fixed Δ, no exploration) in this setting is as in Fig. 6. First, we ran our algorithm on a setting without mood shifts (black line with squares). The algorithm recommended first all available items of the *desirable* type, leaving for the end the *hated* items. On the other hand, we introduced from the 20th to the 40th recommendation (red line with diamonds), a temporary mood shift towards *hated* items. Our algorithm, instantly adapted and switched for 20 recommendations to type a. When the mood shift was over, BYLI-LDA switched again to the type b until the exhaustion of available items, hence recommending the remaining items of type a.

5.3 Discussion

All BYLI-LDA variants exhibit good sequential performance across all experimental settings. We note that our algorithm requires minimum tuning, and recommends items that consistently receive high actual ratings.

Table 3. Depiction of the first few most important topics and their respective words, for a specific user

topic #	**words**
46	time virus future learns troop open search
71	monster creature human giant fire body brain
70	bank rob robbery work police crew kill thief

Our method, in contrast to LSMF (requiring an existing minimum number of recommendations), tackles the cold-start problem by providing personalized recommenda-

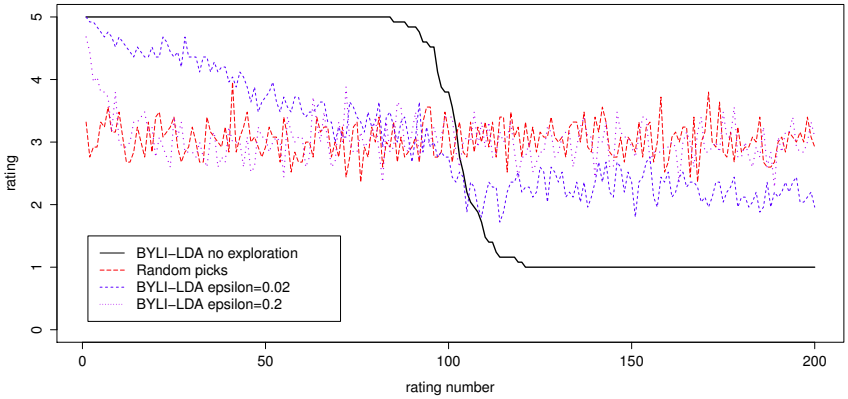

Fig. 5. BYLI-LDA variants vs random

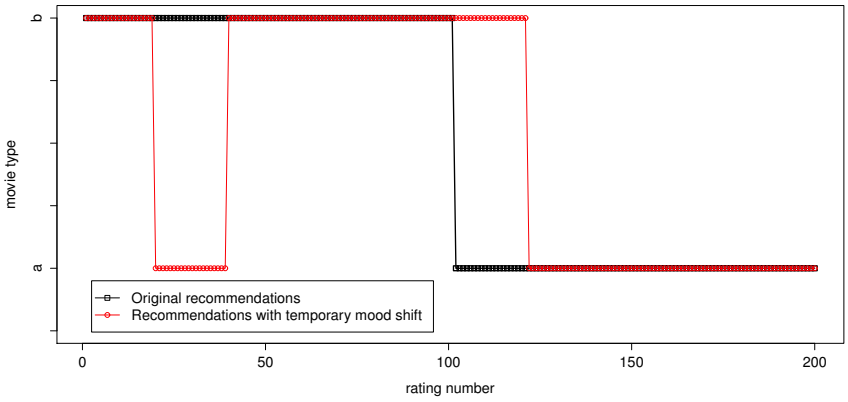

Fig. 6. Behavior when there is a mood shift

tions to a single user, in the absence of any other information. Furthermore, compared to both BYLI and LSMF, our algorithm provides insights about user preferences by providing the topics that describe the user through her item consumption history, as shown in Table 3. The algorithm and infrastructure we developed allow the application of our platform to different domains with minimal effort.

The application of a variable learning rate in conjunction with exploration using a simple method such as ϵ-*greedy* enhances performance, offering further research potential. Another interesting result is the fact that the favoring of the most recent ratings, improves the performance of our algorithm. Intuitively, this could be regarded as an indirect way of boosting exploration by relaxing the adherence of user preferences to dominant topics, across time.

6 Conclusion and future work

In this paper, we employed PTM (in particular LDA) for modeling an evolving user type as a mixture of latent attributes. As such, we offer interpretable recommendations and enhance the intuitions available on each recommendation step. Furthermore, we studied the effects of varying learning rate and a simple exploration approach, during user model updating, with minimal parameter tuning. Our approach is domain agnostic, and avoids the cold-start problem. Moreover, by using a a peer-reviewed, objective corpus, along with its personalized user type updates, it cannot be easily "poisoned" by externally introduced bias. Though it does not surpass state-of-the-art approaches in this particular domain in terms of average recommendation ratings, it exhibits superior sequential performance. For all these reasons, it can be conceivably used as a complementary method to existing state-of-the-art approaches. Future work includes examining the impact of the number of topics K on algorithmic performance, employing more sophisticated RL algorithms, and applying our approach to different domains.

References

1. Blei, David M and Ng, Andrew Y and Jordan, Michael I: Latent dirichlet allocation The Journal of machine Learning research, 3, 993–1022 (2003)
2. Wang, Chong and Blei, David M: Collaborative topic modeling for recommending scientific articles Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, 448–456 ACM (2011)
3. Babas, Konstantinos and Chalkiadakis, Georgios and Tripolitakis, Evangelos: You are what you consume: a bayesian method for personalized recommendations Proceedings of the 7th ACM conference on Recommender systems (221–228) ACM (2013)
4. Koren, Yehuda and Bell, Robert and Volinsky, Chris: Matrix factorization techniques for recommender systems Computer 42, 8, 30–37 IEEE (2009)
5. Melville, Prem and Mooney, Raymond J and Nagarajan, Ramadass: Content-boosted collaborative filtering for improved recommendations AAAI/IAAI, 187–192 (2002)
6. Mooney, Raymond J and Roy, Loriene: Content-based book recommending using learning for text categorization Proceedings of the fifth ACM conference on Digital libraries, 195–204 ACM (2000)

7. Bowling, Michael and Veloso, Manuela: Rational and convergent learning in stochastic games Proceedings of the 17th International Joint Conference on Artificial Intelligence, Volume 2 1021–1026 (2001)
8. Bowling, Michael and Veloso, Manuela: Multiagent learning using a variable learning rate, Artificial Intelligence, 136, 2, 215–250 Elsevier (2002)
9. Sutton, Richard S and Barto, Andrew G: Introduction to reinforcement learning, MIT Press (1998)
10. Koren, Yehuda: The bellkor solution to the netflix grand prize, Netflix prize documentation, 81, (2009)
11. Piotte, Martin and Chabbert, Martin: The pragmatic theory solution to the netflix grand prize, Netflix prize documentation (2009)
12. Toscher, Andreas and Jahrer, Michael and Bell, Robert M: The bigchaos solution to the netflix grand prize, Netflix prize documentation (2009)
13. Langseth, Helge and Nielsen, Thomas Dyhre: A latent model for collaborative filtering, International Journal of Approximate Reasoning, 53, 4, 447–466, Elsevier (2012)
14. Zhang, Yi and Koren, Jonathan: Efficient bayesian hierarchical user modeling for recommendation system, Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 47–54 ACM (2007)
15. Bresler, Guy and Chen, George H and Shah, Devavrat: A Latent Source Model for Online Collaborative Filtering, Advances in Neural Information Processing Systems, 3347–3355 (2014)
16. Hofmann, Thomas: Latent semantic models for collaborative filtering, ACM Transactions on Information Systems (TOIS), 22, 1, 89–115 ACM (2004)
17. Hu, Yifan and Koren, Yehuda and Volinsky, Chris: Collaborative filtering for implicit feedback datasets, Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, 263–272 IEEE (2008)
18. Blei, David M: Probabilistic topic models, Communications of the ACM, 55, 4, 77-84 ACM (2012)
19. Koren, Yehuda and Bell, Robert: Advances in collaborative filtering, Recommender systems handbook, 145–186 Springer (2011)
20. Agarwal, Deepak and Chen, Bee-Chung: fLDA: matrix factorization through latent dirichlet allocation, Proceedings of the third ACM international conference on Web search and data mining, 91–100 ACM (2010)
21. Ling, Guang and Lyu, Michael R and King, Irwin: Ratings meet reviews, a combined approach to recommend, Proceedings of the 8th ACM Conference on Recommender systems, 105–112 ACM (2014)
22. Kurimo, Mikko: Indexing audio documents by using latent semantic analysis and som, Elsevier (1999)
23. Wallach, Hanna M and Murray, Iain and Salakhutdinov, Ruslan and Mimno, David: Evaluation methods for topic models, Proceedings of the 26th Annual International Conference on Machine Learning, 1105-1112 ACM (2009)
24. Hoffman, Matthew and Bach, Francis R and Blei, David M: Online learning for latent dirichlet allocation, advances in neural information processing systems, 856–864 (2010)
25. McCallum, Andrew Kachites: MALLET: A Machine Learning for Language Toolkit, http://mallet.cs.umass.edu (2002)