Multiple Domain User Personalization

Yucheng Low
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ylow@cs.cmu.edu

Deepak Agarwal
Yahoo! Research
Santa Clara, CA 95051, USA
dagarwal@yahoo-inc.com

Alexander J. Smola
Yahoo! Research
Santa Clara, CA 95051, USA
alex@smola.org

ABSTRACT

Content personalization is a key tool in creating attractive websites. Synergies can be obtained by integrating personalization between *several* internet properties. In this paper we propose a hierarchical Bayesian model to address these issues. Our model allows the integration of multiple properties without changing the overall structure, which makes it easily extensible across large internet portals. It relies at its lowest level on Latent Dirichlet Allocation and on latent side features for cross-property integration. We demonstrate the efficiency of our approach by analyzing data from several properties of a major internet portal.

Keywords

Latent Dirichlet Allocation, User Profiling, Domain Integration, Parallel Statistical Inference

1. INTRODUCTION

The rapid growth of the World Wide Web results in a tremendous amount of information available to people. To prevent information overload, many collaborative filtering and recommendation systems [1, 19, 8, 3, 2] have been developed to filter the information stream and display only information the user may be interested in. Such methods have been successfully deployed in various settings ranging from movies (Netflix) to music (iTunes, last.fm) and books (Amazon).

A problem with existing recommendation systems is that they generally require some amount of user interaction to be recorded before personalization is possible.[1] For instance, a movie recommendation system will need a user to provide preference information for some movies, before it is possible to provide personalized recommendation of other movies; likewise, a news portal will require a number of page views before an interest profile of the user can be constructed. In the meantime, the user will experience poor recommendation results. This is known as the **cold-start** problem. Many methods have been developed to tackle this problem typically by making use of additional information about the user (such as age, gender and occupation) [2]. A variation of the cold-start problem is where a new item is added to the system. In this case there is typically a decent amount of prior information available about the new item and other feature based methods have been developed that performs well [13, 17].

[1]User could be either a browser cookie or registered Yahoo! id. No personally identifiable user information was available in the data analyzed.

In this paper, we propose a solution to the cold-start problem by combining user information across multiple domains(properties). We construct a hierarchical Bayesian model which integrates recommendation systems over multiple different domains (such as the frontpage of an internet portal and its news site) and makes use of the user's interactions in one set of domain to immediately provide a personalized experience in other domains the user may have never interacted with. Note that the domains may operate over different "vocabularies". For instance, this model could be used to infer that a person interested in sports news (at a news webpage) is also interested in buying sports apparel (at a store) even if no metadata matching is provided for the store items. Our contributions include:

- A novel graphical model formulation of the user personalization problem which is easily *extensible* and allows integration over an unlimited number of domains.
- A scalable inference procedure which allows the model to scale to millions of users.
- Demonstrating excellent predictive performance on a large real-world dataset.

Outline: We begin with a background overview of related hierarchical graphical models used for text analysis. Next, we provide a detailed description of the estimation and the model associated with it in Sections 3 and 4. This is followed in Section 5 by details on how to implement the estimator efficiently. Experimental results are reported in Section 6 and we conclude with a discussion and an outlook towards more advanced user modeling techniques.

2. BACKGROUND

Latent variable graphical models have found great success in text analysis applications; of which Latent Dirichlet Allocation (LDA) [4] is probably the most well known model. we provide a brief overview of LDA here since our proposed model has LDA as a sub-model.

2.1 Latent Dirichlet Allocation

Latent Dirichlet Allocation [4] aims to extract "semantically valid" topics from document collections. Each topic is simply a distribution over words in a fixed vocabulary. For instance, Table 2.1 gives a few plausible examples of LDA topics. The size of the word represents the probability of the word in the topic.

Topic 1	basketball NBA hoop train 3-point
Topic 2	golf hole-on-one Tiger Woods club
Topic 3	learning network latent machine neural train

Table 1: Example of LDA topics. Size of word represents the probability of the word in the topic.

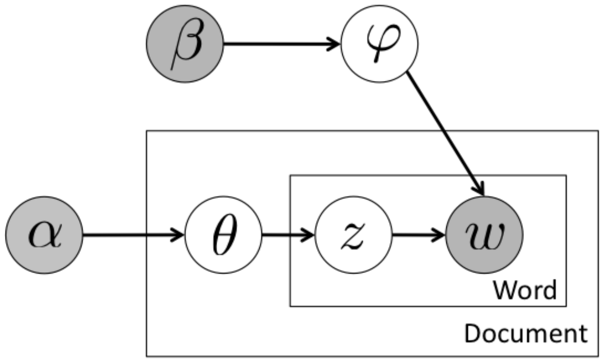

Figure 1: LDA Bayes Net

In LDA, each document is represented as "bag of words" where the sequence of words, as well as the number of occurences of each word, is ignored. The aim of LDA is then to infer for each document, what is its distribution over topics (is this document about both basketball and golf? Or is this document about machine learning?), as well as the contents of each topic.

Letting φ_t be the word distribution of topic t, the generative procedure is as follows:

1. For each topic t:
   (a) Generate the topic->word distribution $\varphi_t \sim \mathcal{D}(\beta)$
2. For each document i:
   (a) generate its distribution over topics $\theta_i \sim \mathcal{D}(\alpha)$
   (b) For each word $w_{i,j}$ in document i:
      i. Pick the topic of the word: $z_{i,j} \sim \mathcal{MN}(\theta_i)$
      ii. Pick the word: $w_{i,j} \sim \mathcal{MN}(\varphi_{z_{i,j}})$

The equivalent Bayes Net is provided in Fig. 1.

Where α and β are fixed constant priors. Intuitively α is the "smoothness" of document's topics (low α's allow for more variability of topics across documents). While β is the "smoothness" of the topic->word distributions (low β's allow more variability of words across topics).

The typical inference procedure used is the collapsed Gibbs Sampler described in [7] where θ is integrated out. [20] describes an equivalent sampler which is significantly faster through careful choice of data structures and creative rearrangement of the conditional probabilities.

2.2 Dirichlet Multinomial Regression

[11] provides an extension to the LDA model called the Dirichlet Multinomial Regression (DMR) model which allows conditioning on arbitrary document features. For instance, we could learn a topic model which includes the authors of the document as features. This allows us to make predictions of the form "Author X prefers to write documents about topics Y".

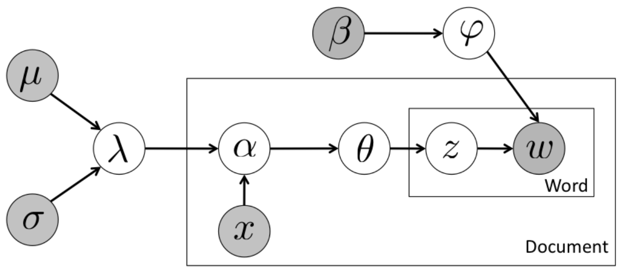

Figure 2: Dirichlet Multinomial Regression

The Bayes Net of the model is provided in Fig. 2. The DMR model extends the LDA model by letting the topic prior α be a parameter which is computed from the document feature vector x. Where λ is a matrix of size #features by #topics,

$$\alpha = \exp(\lambda \times x).$$

Essentially, x_i is a (possibly sparse) document feature vector for document i, and λ is a matrix which projects the feature vector into the "topic-space". The exp is used to ensure positivity of α.

The inference procedure is straight-forward: Fixing λ, the remaining model is identical to LDA, allowing the standard LDA Gibbs sampler to be used on z and φ. λ however is difficult to sample, and [11] optimizes it using L-BFGS.

3. MODEL

3.1 Data Integration

Data integration between different sites is a key problem and challenge for large internet portals. For instance, for a content provider such as Yahoo, Google or Microsoft, serving specialized content sites such as search, news, advertising, a portal, instant messaging, photos, video, cars, dating, etc. it is highly desirable to be able to leverage information from one site on the other. Some of the benefits from such an approach are:

- We can use information gained about a user on one site to provide personalized content right from the beginning on a second site, too. This addresses issues with the cold-start problem.
- We can use such information for improving the overall recommendation accuracy even for users which visit several sites frequently.
- Cross property information helps with abuse detection.
- Personalization can be used for recommending other properties.

Naively we could simply attempt to address the problem by having one joint set of user specific features that are shared by all properties and which are modified based on the user behavior. However, this approach is problematic for a number of reasons:

1. Users may exhibit different site-specific behavior which may only partially relate to the big picture about them.

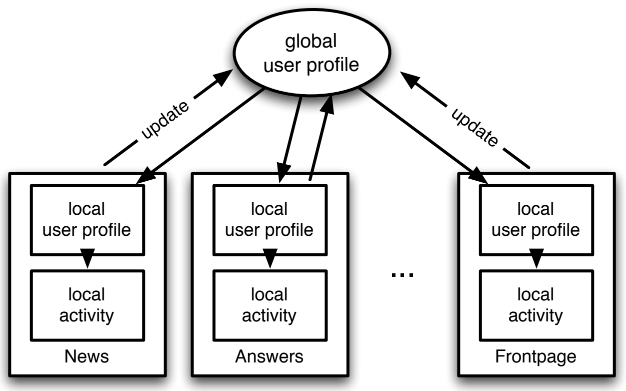

Figure 3: The highlevel structure of the proposed hierarchical statistical model. All properties share (and update) a joint profile vector. In addition to that, each property may keep a local model of the user's behavior in order to adapt to the possibly idiosyncratic activities of a user on a given site.

For instance, a user may well fancy himself as an expert in medieval numismatics on an answering site, yet at the same time he may exhibit some more mainstream preferences on the other sites. This means that we need to *allow* for the possibility of rather diverse and incoherent behavior between sites, and thus additional site-specific personalization, in our model.

2. It must be easy to add sites to the profiling system without the need to re-engineer the model of the already integrated sites. This is a practical imperative since in reality internet properties are launched, bought or sold and our model must accommodate these requirements efficiently.
3. The framework needs to allow for different levels of sophistication in the model across different sites. That is, while some properties might only be willing to use a simple demographic targeting approach, others may be open to using an advanced nonparametric Bayesian approach such as Latent Dirichlet Allocation [4].

In this paper we propose a flexible model that addresses all of these concerns in a principled fashion, namely by means of directed graphical models [14]. Its main structure is given in the diagram of Figure 3.

There are two basic use cases for such a model. In the first use case, each domain already has an existing personalization system. This model could then be used to provide for each user an "initialization prior" allowing personalization on the first visit to a new domain. In the second use case, the existing personalization system can be directly integrated into the hierarchical model, allowing personalization in different domains to be inferred jointly, potentially improving performance. The second use case is of course preferred.

3.2 Formal Definition

In the following we denote by U the set of users in the system and let $|U|$ be the number of users. Furthermore let D be the set of personalization domains, such as News, Answers, Frontpage or Movies. Each user $u \in U$ interacts with one or more domains $d \in D$ producing a sparse list of tokens $W_{du} \subseteq \mathcal{W}_d$. Note that the space $\mathcal{W}_d$ can be different from domain to domain. For instance, in the News domain, this could be the text of the documents viewed, while in the Movies domain, this could be the list of movies rated. Additionally, let D_u denote the set of domains the user interacted with. Note that the number of domains a user interacts with is variable. So is the number of interactions per site.

The task is to predict for each user his preferences in all domains — for those he visited we would like to improve the estimates, for those he never saw before we would like to obtain a good estimate of his future behavior. The latter is clearly the more difficult task as we study the case where we have *no* information regarding a user's behavior on, say, a news site at all. Hence, we will focus on the cold-start problem while noting that the multiple-site personalization problem is contained in it. Formally, for each user $u \in U$, predict W_{du} where $d \notin D_u$.

As measures for the accuracy of the estimator we use a) the increase in log-likelihood by combining several domains, i.e. the increase in statistical prediction accuracy by combining several domains; b) the increase in the cosine similarity score when using several domains for improved personalization. Before we discuss the design of the graphical model associated with Figure 3 let us evaluate some alternatives.

3.3 Alternatives

Joint parameter vector:
Inferring two or more domains jointly is difficult due to the different domain spaces $\mathcal{W}_d$. The simplistic solution of combining all the domains by letting the domain be $\bigcup_{d \in D} \mathcal{W}_d$ is undesirable as it restricts the model excessively. Such a model will not deal well with often rather different amounts of data per domain.
For instance, if we naively combine the Movie and News domains by using article headlines for the News domain, and movie titles for the Movies domain, the model would infer that a person interested in Sports news, must also be interested in Sports movies, and a person interested in Political news must also be interested political documentaries. The model will have difficulties modelling users who are interested in Political news and Action movies since it ignores the fact that users may exhibit different personalities on different properties.
Secondly, if we combine search queries and page views we might end up with several orders of magnitude more tokens in the case of page views. Simply combining both datasets would give queries only a minuscule weight and consequently underestimate their importance as relevant user features.

Joint action space: Another possibility is to combine the domains in a way such that tokens from one domain do not interfere with tokens from the other domain. For instance, this could be done in the Movie and News example by append the numeral "1" to every word in the movie title, and appending the numeral "2" to every word in the news title. Forcing the "dictionary" of both domains to be different, but inferring a joint model naively will resolve the case described in the previous paragraph, but still will not lead to a desirable solution. There are several issues.

- The "model complexity" of the domains could be very different. For instance, the number of tokens in the News domain could be significantly greater than the number of tokens in the Movie domain. Inferring them jointly will cause the latent factors will be largely dominated by the News. Moreover, the problem of recommending movies might be considerably harder than that of recommending news, thus requiring a richer latent representation.

- The modelling complexity could grow rapidly with the number of domains. For instance, we could observe that the News domain alone could be modelled with 10 latent factors, while the Movie domain could be modelled with 5 latent factors. However, when inferred jointly, the number of latent factors required could be much larger than $10+5 = 15$ since there could be one group of users who are interested in Political news and Action movies, as well as another group of users who are interested in Political news and Horror movies. These would have to be modelled as different latent factors, resulting in potentially a multiplicative increase in the number of factors necessary to build an accurate model. This model therefore does not provide the scalability necessary to extend to a large number of domains.

Independent Parameter Vector: This model trivially solves both problems mentioned above, simply by modeling each domain on its own. However, such an approach entire defeats the purpose of integration. In the following we propose a modification of this model to allow for cross-domain personalization.

3.4 Hierarchical Latent Model

Our model resolves these problems by adding an additional layer of latent parameters. We begin by assuming that each user has a hidden **latent feature vector** x_u which completely describes the user's *global* preferences and interests. Each domain also has a hidden **latent domain matrix** λ_d. Furthermore we assume that for each property d the user u also exhibits local traits, say θ_{du}. We assume that the local traits θ_{du} characterize the actions W_{du} that we observe. This yields the following generative model:

1. For all properties d do
   (a) Sample trait vector λ_d
2. For all users u do
   (a) Sample user trait x_u
   (b) For all properties d do
      i. Sample $\theta_{du}|x_u, \lambda_d$
      ii. Sample user actions $W_{du}|\theta_{du}$

In other words, the joint distribution is given by

$$p(\lambda, x, \theta, W) = \prod_d p(\lambda_d) \prod_u p(x_u) \prod_{d,u} p(\theta_{du}|x_u, \lambda_d) p(W_{du}|\theta_{du})$$

This has the advantage that we can express the distribution of the global user vector in terms of its local parameter estimates only. Moreover, the distribution factorizes, that is we have

$$p(x_u|\text{rest}) \propto p(x_u) \prod_d p(\theta_{du}|x_u, \lambda_d). \tag{1}$$

4. STATISTICAL FORMULATION

To make further progress we need to specify the distributions involved in $p(\lambda, x, \theta, W)$. Specifically we use Latent Dirichlet Allocation [4] (LDA) to describe $p(W_{du}|\theta_{du})$ and we subsequently employ a latent variant of "upstream conditioning" [11] to couple properties into a joint graphical model. We begin with an explanation of plain LDA.

4.1 Latent Dirichlet Allocation

One model for estimating user behavior is to assume that users have a number of interests and that their actions are given by a combination of said interests. For instance, if we had a model describing which websites are visited by users interested in politics and which websites are visited by users interested in automobiles then we could easily infer the activity pattern of a user interested in both topics, simply by mixing the topical interest patterns of both.

Such a model is considerably more flexible than, say, an attempt to cluster users: in the latter case we would need to generate clusters for each and every *combination* of interests. Even worse, interests can be expressed to various degrees, so a simple model clustering users as being interested in cars may not be sensitive enough to infer whether users are die-hard car fanatics or whether they just exhibit a fleeting interest in automobiles.

LDA addresses this issue as follows: each user is endowed with a topic mixture θ_u that is drawn from some Dirichlet distribution $\mathrm{Dir}(\alpha)$. Subsequently, for each user action, we draw a topic z_{ui} from the user-specific topic distribution θ_u and finally, we draw an action w_{ui} from the action distribution associated with topic z_{ui}. The action distributions φ_k themselves are drawn from a Dirichlet distribution. The parameters α and β are priors that adjust how nonuniform the topic distributions and the word distributions are. We obtain [4]

$$\begin{aligned} & p(w, z, \theta, \varphi|\alpha, \beta) \\ = & \prod_k p(\varphi_k|\beta) \prod_u p(\theta_u|\alpha) \prod_{i \in W_u} p(z_{ui}|\theta_u) p(w_{ui}|z_{ui}, \varphi) \end{aligned} \tag{2}$$

Large scale inference for (2) presents a significant challenge and there are only few published results on how to apply LDA to millions of instances (i.e. users). We use the approach of [18] for distributed collapsed inference and build on this framework. Before going into details regarding large scale estimation let us extend (2) in the spirit of Figure 3.

4.2 Upstream Coupling

The key to combining different properties is to make the prior α depend on both the property and the global features of the user. That is, instead of fixing α we assume that $\alpha_{du}|x_u, \lambda_d$ where x_u is the global user feature and λ_d is the site specific trait.

More specifically, to interact with a domain, a nonnegative[2] **user-domain feature vector** α_{du} is produced by computing the logistic transfer function $\mathrm{lgt}(\lambda_d x_u)$ where

$$\mathrm{lgt}(y) = \log\left[1 + e^y\right]. \tag{3}$$

In other words, α_{du} is computed by multiplying the user's **latent feature vector** with the domain's **latent domain**

[2]Nonnegativity of the coefficients α is needed to encode a valid Dirichlet prior on the topic distribution.

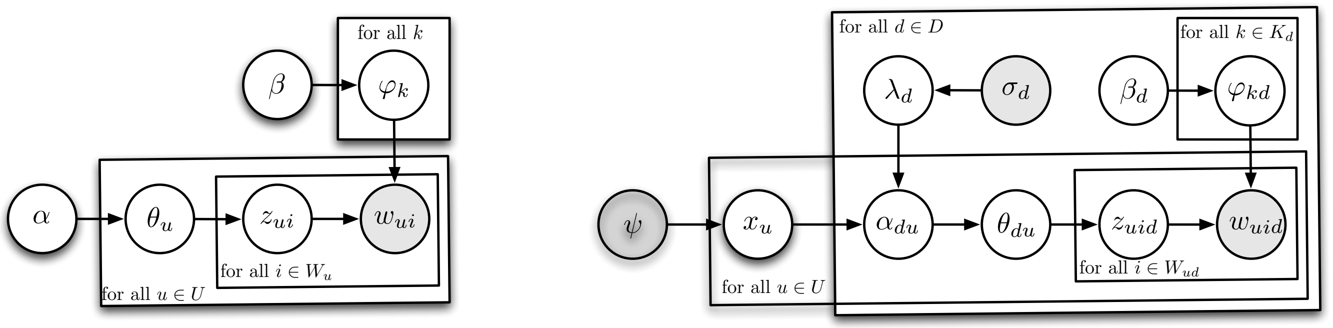

Figure 4: Left: Latent Dirichlet Allocation. This model covers the behavior of users u, as expressed in the actions w_{ui} on a single property. Right: Hierarchical Model with LDA applied to all properties. Each property has a characterization vector λ_d and each user is endowed with a global personalization vector x_u. σ_d and ψ are priors for these personalization terms.

matrix and passed through a non-linear transform to enforce positivity. Note that the choice of lgt is somewhat arbitrary as there are many ways to project from $(-\infty, \infty)$ to $[0, \infty)$. In [11] this is done by exponentiation. However, we found exponentiation to be difficult to work with as its steep gradients cause numeric instabilities in inference. Hence we chose lgt for its simplicity. If sparsity in α is desirable, other possibilities such as Huber or L1 loss can be considered.

The **user-domain feature vector** is a description of how the user may interact with a given domain and can be interpreted freely by the domain specific model. As already outlined above we use LDA [4] as the domain specific model. However note that the design of Figure 3 does not require this assumption.

Essentially, the user's **latent feature vector** can be interpreted as a "super-topic" in the same spirit as [12]. Each column in the **latent domain matrix** can then be interpreted as the distribution of topics inside the super-topic. Multiplying these two latent parameters together will therefore produce a distribution over topics within the domain, which can then be used as a prior (α_{du}) for the actual topic distribution of the user's interaction within the domain.

Note that this model allows us to have different degrees of complexity in different domains: while the dimensionality of the global latent space is fixed via $x_u \in \mathbb{R}^k$ we may have different numbers of topics on different properties via $\lambda_d x_u$, simply by choosing different dimensions for different λ_d. This makes sense from a modeling perspective — some properties may only see small amounts of a user's activity (e.g. it is unlikely that on a cars site we may care much about the interest of a user in medieval numismatics). In summary the generative process is as follows:

1. For all properties d do
   (a) Generate **latent domain matrix** $\lambda_d \sim \mathcal{N}(0, \sigma_d)$
2. For all users u do
   (a) Generate **latent feature vector** $x_u \sim \mathcal{N}(0, \psi)$
   (b) For all visited properties $d \in D_u$ do
      i. Let **user-domain feature vector** be
      $$\alpha_{du} = \operatorname{lgt}(\lambda_d x_u)$$
      ii. Draw user-domain topic distribution
      $$\theta_{du} \sim \mathcal{D}(\alpha_{du})$$
      iii. for each observed token $i \in W_{du}$
         A. Draw topic $z_{dui} \sim \mathcal{MN}(\theta_{du})$
         B. Draw word $w \sim \mathcal{MN}(\phi_{z_{dui}})$

The graphical model is in Fig. 4. We observe that if all user and domain latent parameters are known, i.e. if λ and x are observed, each domain essentially reduces to an instance of LDA. This therefore provides a relatively straightforward inference scheme.

4.3 Inference

Making the observation that once all λ and x are fixed, the model is essentially equivalent to having $|D|$ independent LDA models, where users are documents, and where the topic prior is different for each document. We can therefore perform collapsed sampling in a similar manner as in [7] by integrating out θ and φ, and sampling z. In addition we sample x using Langevin diffusion. The λ parameter is optimized using L-BFGS [9]. We describe (the rather technical) details below. They are not essential to understanding the experiments but they matter for the purpose of reproducibility of the results. After integrating over θ the complete likelihood $P(z, \lambda, x)$ of the model is given by

$$\prod_{d,u:d\in D_u} \frac{\Gamma(\sum_{t_d} \operatorname{lgt}(x_u^T \lambda_{d,t_d}))}{\Gamma(\sum_{t_d} \operatorname{lgt}(x_u^T \lambda_{d,t_d}) + n_{du})} \prod_{t_d} \frac{\Gamma(\operatorname{lgt}(x_u^T \lambda_{d,t_d}) + n_{t|du})}{\Gamma(\operatorname{lgt}(x_u^T \lambda_{d,t_d}))} \tag{4}$$

$$\prod_{d,t_d,k} \frac{1}{2\pi\sigma_k^2} \operatorname{lgt}\left(\frac{-\lambda_{d,t_d,k}^2}{2\sigma_k^2}\right) \prod_{u,k} \frac{1}{2\pi\psi_k^2} \operatorname{lgt}\left(-\frac{x_{u,k}^2}{2\psi_k^2}\right)$$

Note that this is very similar to [11, Eq. (1)], but with user-properties instead of documents, and different choices for λ and x. Let $\operatorname{dlgt}(y) := \partial_y \operatorname{lgt}(y)$ be the derivative of the logistic transfer function. In this case the derivative of the log likelihood with respect to $\lambda_{p,t_p,k}$, i.e. $\partial_{\lambda_{p,t_p,k}} \log P(z, \lambda, x)$ is given by

$$-\frac{\lambda_{d,t_d,k}}{\sigma_k^2} + \sum_{u:d\in D_u} x_{u,k}\,\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)\times \tag{5}$$
$$\left[\Psi\left(\sum_{t_d}\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)\right) - \Psi\left(\sum_{t_d}\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)+n_{du}\right)\right.$$
$$\left.+\Psi\left(\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)+n_{t|du}\right)-\Psi\left(\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)\right)\right]$$

We use this equation to optimize λ using L-BFGS. The derivative $\partial_{x_{u,k}}\log P(z,\lambda,x)$ with respect to $x_{u,k}$ is similar:

$$-\frac{x_{u,k}}{\psi_k^2} + \sum_{u:d\in D_u}\sum_{t_d}\lambda_{p,t_d,k}\,\text{dlgt}\left(x_u^T\lambda_{p,t_d}\right)\times \tag{6}$$
$$\left[\Psi\left(\sum_{t_d}\text{dlgt}\left(x_u^T\lambda_{p,t_d}\right)\right) - \Psi\left(\sum_{t_d}\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)+n_{du}\right)\right.$$
$$\left.+\Psi\left(\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)+n_{t|du}\right)-\Psi\left(\text{dlgt}\left(x_u^T\lambda_{d,t_d}\right)\right)\right]$$

This equation allows us to sample each $x_{u,k}$ using a Langevin Diffusion sampler.

4.4 Langevin Sampler

The Langevin Diffusion sampler [15] is essentially a Metropolis-Hastings sampler with a Gaussian proposal shifted in the direction of the gradient: Where the current value of $x_{u,k}$ is $x_{u,k}^t$ (sample value at time t), we propose the new value:

$$y_{u,k} = x_{u,k}^t + \frac{\sigma^2}{2}\frac{\partial\log P(z,\lambda,x)}{\partial x_{u,k}}\bigg|_{x_{u,k}^t} + N(0,\sigma^2)$$

Where the notation $\frac{\partial\log P(z,\lambda,x)}{\partial x_{u,k}}|_{x_{u,k}^t}$ means to take the partial derivative of $\log P(z,\lambda,x)$ against the variable $x_{u,k}$ (Eq. (6)) and evaluate it at the value $x_{u,k}^t$.

We then compute the acceptance ratio:

$$r = \frac{\exp\left\{-\left\|y_{u,k}-x_{u,k}^t-\frac{\sigma^2}{2}\frac{\partial\log P(z,\lambda,x)}{\partial x_{u,k}}\bigg|_{x_{u,k}^t}\right\|^2 \Big/ 2\sigma^2\right\}}{\exp\left\{-\left\|x^t{}_{u,k}-y_{u,k}-\frac{\sigma^2}{2}\frac{\partial\log P(z,\lambda,x)}{\partial x_{u,k}}|_{y_{u,k}}\right\|^2 \Big/ 2\sigma^2\right\}}$$

Then update $x_{u,k}^t$ to the proposed value $y_{u,k}$ with probability $\min(r,1)$:

$$x_{u,k}^{t+1} = \begin{cases} y_{u,k} & \text{with probability } \min(r,1) \\ x_{u,k}^t & \text{with probability 1-}\min(r,1)\end{cases}$$

5. IMPLEMENTATION

To run experiments on large datasets, it is necessary to make use of large scale distributed processing. We therefore implemented the model by extending the distributed LDA implementation from [18]. The extension is relatively straightforward due to the similarities to regular LDA. The main contributions we made to the implementation is a distributed L-BFGS solver, as well as a novel way to initialize the distributed sampler which accelerates convergence significantly.

Algorithm 1 Sampling executed on each machine.

```
for Each user $u$ in this machine's partition do
  Initialize $u$'s latent feature vector
  for Each domain $d \in D_u$ the user interacted with do
    for Each observed token $w \in W_{du}$ do
      Sample $w$'s topic using the procedure in Sec. 5.2
    end for
  end for
end for
if I am Machine 0 then
  Initialize latent domain matrices
  Store latent domain matrices into Memcached
else
  Initialize latent domain matrices from Memcached
end if
for $i = 1$ to #sampling iterations do
  if $i$ mod $15 = 0$ then
    \\ Optimize every 15 iterations
    Optimize latent domain matrices using
        distributed L-BFGS
  end if
  for Each user $u$ in this machine's partition do
    Sample $u$'s latent feature vector using
        Langevin Diffusion
    Sample the topic in each observed token in $u$
  end for
end for
```

An overview of our approach is provided in Algorithm 1. Firstly, the set of users are equally partitioned among the P machines in the cluster. Each machine then initializes each user with a random **latent feature vector**, and uses the Distributed Initialization procedure described in Sec. 5.2 to provide the initial topic assignments to each observed token, producing local word-topic counts. A set of `memcached` servers[3] are used to synchronize the local word-topic counts with the global word-topic counts. See [18] for details on the communication between nodes. The **latent domain matrix** is initialized by having one machine generate the random matrix, writing it into `memcached`, and all remaining machines reading from it.

The sampler then proceeds as described in [18], with the exception that the user's latent feature vector must be sampled, and the latent domain matrices are optimized periodically (every 15 iterations in the pseudo-code in Alg. 1).

5.1 Distributed L-BFGS solver

The latent domain matrices are optimized using a distributed L-BFGS solver. We first observe that the gradient of the log-likelihood of the model with respect to the domain matrices λ can be written as a sum over all the users in the dataset. This naturally means that the gradient can be computed in a fully distributed fashion, collected and summed on one machine, which then performs the actual L-BFGS update. This procedure is however further complicated by the Armijo line-search stage of the L-BFGS solver, which requires the log-likelihood and and the gradient to be recomputed repeatedly for different values of λ. This requires a close level of coordination among all the machines which

[3]`memcached` is a fast distributed (key, value) storage system

is not immediately provided by the `memcached` framework.[4]

To implement this algorithm, we must first re-implement three standard distributed programming primitives within the memcached environment. The three primitives are the **barrier**, **sum** and **broadcast**.

The **barrier** is a function which provides "sequentialization" of execution across a distributed program. When a process call the **barrier** function, the process will stop execution and pause. The **barrier** function only returns when all machines in the distributed system enter the **barrier**. A common use-case for the **barrier** is to "wait" for all processors to complete a particular task, before going on to the next task. We implement a version of the sense-reversing barrier algorithm described in [10].

The distributed **sum** function (or more generally also known as **reduce** or **fold**) simply collects and aggregates a value posted by each machine. Each machine p calls the **sum** function with a particular value v_p. The function then returns with the sum of all the values $(v_1 + v_2 + \cdots + v_{P-1} + v_P)$. We implement this operation by having each machine write its value v_p to a `memcached`. A **barrier** is then called to guarantee that all machines have completed the write. Machine 1 then reads all the values from `memcached`, sums them and writes back to `memcached`. Another **barrier** is used here to ensure that the write is completed. Then all machines reads the resultant value from from `memcached`.

The **broadcast** operation allows one machine to send a value to all the other machines in the network. This operation is similar to the **sum** operation.

Combining these operations, we can write the distributed L-BFGS solver in pseudo-code in Alg. 2.

Algorithm 2 Pseudo-code of Distributed L-BFGS Solver ran on each machine in the cluster

```
\\ compute the initial gradient and log likelihoods
Compute the local gradient $g$ Using Eq. (5)
Compute the local log likelihood $l$ Using Eq. (4)
Perform distributed sum over $g$ and $l$
if I am machine 1 then
    Compute L-BFGS step $\Delta$ using $g$ and $l$
    Update $\lambda$
    broadcast $\lambda$
else
    receive broadcast $\lambda$
end if
\\ Armijo Line search
while True do
    Compute the local gradient $g'$ Using Eq. (5)
    Compute the local log likelihood $l'$ Using Eq. (4)
    Perform distributed sum over $g'$ and $l'$
    if I am machine 1 then
        Test Armijo and Wolfe conditions using $g'$ and $l'$
        \\ quit if armijo conditions are satisfied
        Let quit = 1 if Armijo conditions are satisfied
        Let quit = 0 otherwise.
        broadcast quit
        if quit then
            break
        else
            \\ conditions not satisfied. Perform line search
            Decrease step-size and update $\lambda$
            broadcast $\lambda$
        end if
    else
        receive broadcast quit
        if quit then
            \\ If quit is true then Armijo conditions are satisfied
            break
        else
            \\ If not, line search proceeds
            \\ and we need to receive the new $\lambda$
            receive broadcast $\lambda$
        end if
    end if
end while
```

5.2 Distributed Initialization

The standard way to initialize an LDA sampler is to sample the topic of each new word from a uniform distribution. However, this results in slow convergence. A more effective way of initializing the sampler is to perform online sampling [5]. That is, the topic of each new word is sampled by conditioning on all the words seen so far. This procedure is easy when performed on a single machine, but is extremely complex to implement in a distributed setting as the global topic counts will need to be synchronized very frequently.

A plausible alternative is to have each machine perform the online sampling process independently using only the local topic counts. This however, produces poor initializations as each machine could assign very different topics to the same word resulting in a uniform topic distribution when the topic counts are aggregated across all the machines.

Instead, we propose a simple solution: We perform online sampling independently on each machine as described above. However, when we sample a topic for a new word w, instead of using a random/pseudo-random generator, we use $hash\,(w + \#\text{occurrences of w seen so far})$ as the random number.

This procedure is equivalent to constructing a different pseudo-random generator for each unique word, and that this generator is seeded identically across all the machines. This algorithm does not correspond to a true online-LDA initalization, but it simply "correlates" the random number generator across all machines, encouraging similar topic distributions for each word. An advantage of this procedure is that it does not require any communication between the machines, allowing for a simple implementation.

[4]One might suspect that the entire algorithm could be efficiently implemented in Hadoop MapReduce. However, one of the issues is that by deferring all inter-process communication to the Reduce phase convergence of the sampling algorithm is severely hampered. Moreover, it creates artificial barriers to the sampling iterations which can lead to considerable performance penalties. For a detailed discussion see [18].

6. TWO DOMAIN EXPERIMENTS

We tested the model on a large real world dataset comprising a random subsample of 5.6 million users who interacted with Yahoo! News and Yahoo! Frontpage for a week. We record "click-through" information for each user. That is to say, when the user clicks on a Frontpage link, we record the words in the link. Likewise, when the user clicks on a News article, we record the words in the article.

The goal is to predict how a new user might interact with

Frontpage, given the user's past interactions with News (and vice versa). The problem is made particularly difficult since only 1% of all the users in our dataset interacted with both Yahoo! News and Yahoo! Frontpage.

To perform holdout testing, we selected 20% of the users who interacted with both domains and hide the user's interaction with one of the properties when training the model. The model can then be used to predict the user's interaction with the hidden property. Where α_{du} is the user-domain feature vector, we note that the predicted word distribution $P(w_{du}|\alpha)$ is simply proportional[5] to $\varphi \times \alpha_{du}$. We evaluate accuracy using the cosine similarity score between the predicted word distribution $P(w_{du}|\alpha)$ and the true observed word distribution w_{du}.

$$\frac{\langle w_{du}, \varphi \times \alpha_{du} \rangle}{|w_{du}| \times |\varphi \times \alpha_{du}|}$$

We tested against a simple baseline, which is the mean-prediction formed by averaging the word distribution across all the user-interactions within the domain. This intuitively represents the "average" user. We do not test against nearest-neighbor methods as they require large amounts of computation to evaluate each new user and do not scale well to large dataset sizes. We also do not consider the method of combining all the domains into a single LDA model (joint parameter vector and joint action space in Sec. 3.3) to be viable models due to the significant difference in the domain types for the three-domain experiment of Section 7.

Using 100 topics to model FrontPage and 400 topics to model News, we plot in Fig. 5 the model log-likelihood $P(w, z)$ across sampling iterations. We compare the likelihood curves obtained when varying the dimensionality of the latent feature vector. As expected, running independent LDA on both domains (equivalent to using a latent feature vector of length 0) produces the lowest log-likelihood value. Increasing the dimensionality of the latent feature vector from 0 to 25 allows the model to fit the data better, resulting in a higher log-likelihood. Further increasing from 25 to 50 still produces a small increase in log-likelihood, though we observe diminishing returns.

Next in Fig. 6 we plot the % improvement in hold-out accuracy over baseline prediction. We observe significant gains in holdout accuracy for both Front Page and News. Increased model complexity (by increasing the number of latent features) appears to overfit slightly, resulting in decreased hold out accuracy.

Fig. 6 also suggests that it is easier to predict FrontPage preferences from News preferences than vice versa. This is expected since the size of News vocabulary is about 4 times larger than the size of the FrontPage vocabulary. Each interaction with a News article also produces significantly more tokens. Moreover, the range of stories on the FrontPage is considerably more constrained than the set of news events in general. Hence there is simply less information to be gained from knowing which FrontPage story a given user viewed.

Next, since $\lambda_{\text{news}} x_i$ is the topic "preference" for user i in the News domain, and $\lambda_{\text{fp}} x_i$ is the topic "preference" for user i in the Frontpage, we can use $\sum_x (\lambda_{\text{news}} x)(\lambda_{\text{fp}} x)^T$ to estimate the correlation between the topics across News and Frontpage. Using this, we can try to infer that for a person who is interested in a particular topic on Frontpage, what other topics might he be interested in when he goes to the News domain.

[5]This simple identity holds for Dirichlet priors in the absence of any additional data since it forms a conjugate prior to the multinomial mixture.

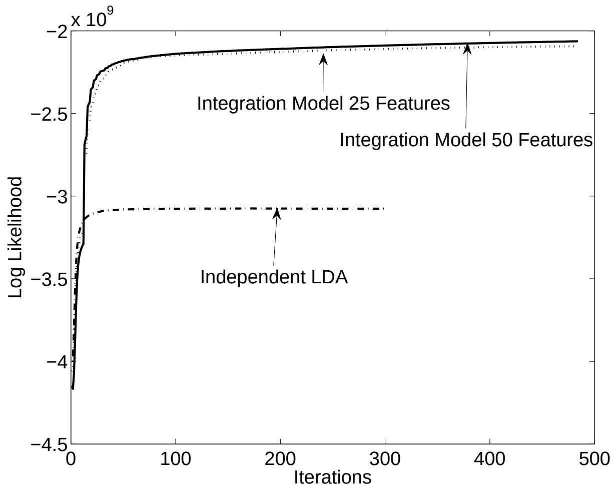

Figure 5: Two Property FrontPage + News model log- likelihood over iterations using 100 topics for FrontPage and 400 topics for News. Each curve corresponds to using a user latent feature vector of different lengths. The "Independent LDA" is the log-likelihood of running LDA independently on both domains and is equivalent to using a latent feature vector of length 0. As expected, increased dimensionality of the latent feature vector results in increased log-likelihood.

A minor problem with this procedure is that it can be hard to factor out strong "background preferences". That is, a Frontpage topic which **everyone** is interested in, might appear to be strongly correlated with almost all other News topics. However, we can still obtain very interpretable results.

In Table 2, we demonstrate this for two random selected topics in Frontpage, and their top correlated topic in News. For instance, the first row suggests that people interested in reading about science articles on Frontpage might also be interested in reading about Avatar (a science fiction movie) on the News website. And people interested in the Oscars might also be interested in reality TV shows.

In Table 3, we demonstrate the reverse. For two randomly selected topics in News, what are their top correlated topics in Frontpage. The first row suggests that people interested in technology, might also be interested in college graduate earnings. In the second row, we encounter the strong "background preference" problem as stated earlier. Here we try to infer the Frontpage preferences of a user who is interested reading about the healthcare debate on News. If we simply look at the top correated topic, we might infer that the user is interested in reading about sports. However, including the second and third most correlated topics, we also see that the user might also be interested in banking and terrorism related articles.

7. THREE DOMAIN EXPERIMENT

FrontPage	News
bacteria, fight, super, struggling, developed, doctors, resistant, lethal, virtually, drugs, antibiotic, competitors, chad, andrews, ochocinco, erin, whos, aas, batteries, stronger	film, movie, movies, films, director, story, avatar, james, time, hollywood, big, make, hes, star, good, remake, horror, great, award, man
sandra, oscar, oscars, red, carpet, bullock, golden, gown, bullocks, nominee, bestactress, sparkles, stunning, retirement, saving, cost, taxes, expenses, fully, major,	vienna, bachelor, jake, pavelka, giraldi, finale, show, stars, dancing, love, season, time, abc, episode, tonights, question, popped, relationship, maintains, man,

Table 2: Given a particular frontpage topic (left column), the top correlated news topic (right column). For instance, the first row suggests that people interested in science might also be interested in Avatar (a science fiction movie).

News	Frontpage
iphone, apple, app, apps, ipod, google, store, apples, android, mac, mobile, touch, ipad, device, phone, screen, jobs, developers, iphones, time,	college, year, earn, years, 000, bestpaid, average, 129, colleges, graduates, ten, alums, schools, actor, likes, prompt, spirit, wrench, time, ghost,
health, care, bill, obama, president, rep, house, republican, senate, news, sen, democrats, fox, congress, reform, federal, majority, obamas, roberts, john	drafts, player, nfl, scouts, team, riskiest, peril, bryant, dez, pick, talented, nba, james, news, 23, familiar, number, lebron, cleveland, decision, home, bank, facing, ceo, gomez, eviction, rosalina, bought, cleaning, foreclosed, client, janitor, offices, surprising, video,, captured, inside, mountain, terrorist, observers, impresses, alqaidas, complexity, base, features, hideout, size, special, secret, struck, sell, products, month, ways, arctic

Table 3: Given a news topic (left column), the top correlated frontpage topic (right column). For the second row, the top three topics are provided.

We next extend the model further by including a third domain: MyYahoo. Here, we demonstrate the scalability and extensibility of the model. To our knowledge this is first attempt at user personalization across three different domains.

For the myYahoo domain, we only record the **id's** of the articles the user clicked on. This makes the problem significantly harder as the article id's have no semantic significance. We selected 5.6 million users from the same date range as the two domain experiment. About 5.5% of the users interact with two or more domains. We selected 10% of the users who interacted with two or more domains for holdout testing.

We trained the model on the dataset, using 50 latent features and plot the % improvement in holdout accuracy over baseline prediction in Fig. 7. We demonstrated improvement for both Frontpage and News, as well as an insignificant improvement in MyYahoo. This is to be expected due to the lack of semantic information in the MyYahoo domain and due to the comparatively small number of applications provided on MyYahoo.

Next, in Fig. 7, we compare the holdout accuracy for FrontPage and News between the 3-domain model and the 2-domain model. Each model was tested with 25, 50 and 75 latent features and the best setting for each model was used. The 2-domain model required only 25 latent features and increasing the feature count further results in overfitting, while the 3-domain model continues to improve in performance up to 50 latent features. We observe that even with the limitations of the MyYahoo data, including the third domain still improves holdout performance significantly on Frontpage prediction (while incurring a small drop for News).

8. EXTENSIONS

The model is designed to be a highly **generic** model for user personalization, supporting a variety of possible extensions.

Firstly, the design of the model does not require all the user features x to be latent. Instead, we may combine our model with the upstream conditioning of [11] which immediately allows us to add observed variables. For instance, one could use geographical location (via the IP address or via HTML5) as a fully observed user feature. Alternatively, the model also permits the use of features which are not observed for all users (such as gender). The latter case is particularly interesting as the model would then try to infer the feature value for the remaining users. For instance, the model would try to infer the gender of user if the user's gender is not observed.

Next, the model does not require the use of LDA within a domain. The inference process is highly modular with regards to each domain, and permits any domain model to be used as long as a log-likelihood derivative with respect to x can be computed.

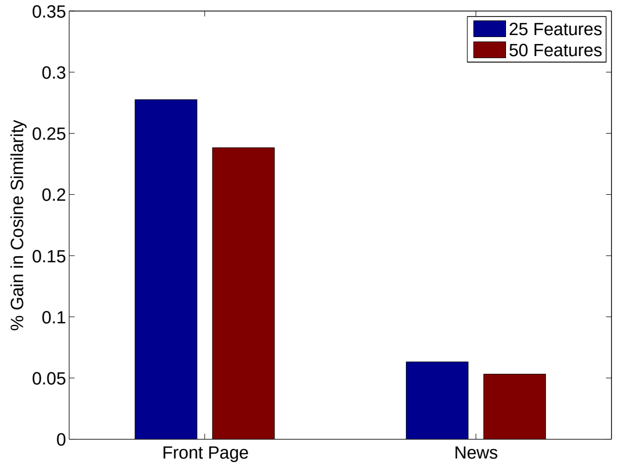

Figure 6: Two Property FrontPage + News Hold out accuracy improvement over baseline for different number of latent features.

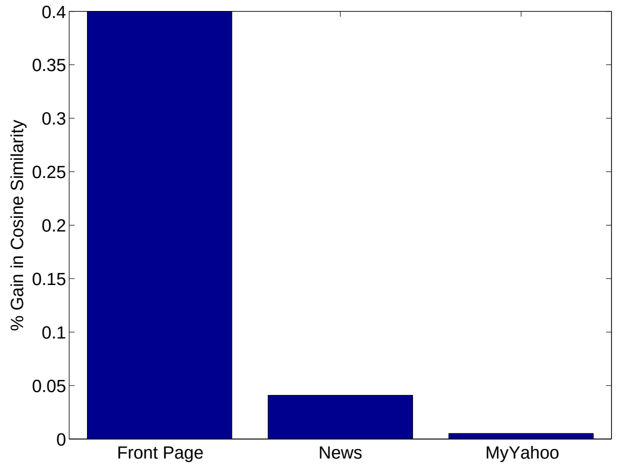

Figure 7: Three Property FrontPage + News + myYahoo hold out accuracy.

- This could permit the model to provide cross-domain personalization into unusual domains such as personalized spam filtering.
- We could use a Gaussian latent matrix factorization model along the lines of [16] which replaces the discrete set of topics by a continuous (low dimensional) factorization.
- We could employ a factorization akin to the Indian Buffet Process [6] which uses a binary latent variable representation instead of a finite (sparse) set of topics.

Also, since the model places no restriction on the number of domains it can personalize over, the model could, with little effort, be used as a back-end to connect other generative user-personalization schemes.

9. CONCLUSION

In this work, we designed a new model for providing user personalization across two or more domains and demonstrated its efficacy on a large real-world dataset comprising of data from Yahoo News, Yahoo FrontPage and MyYahoo, obtaining significant gains in prediction accuracy. The model we developed is **highly extensible** and observed user features such as geographical location and gender can be integrated. Furthermore, the model treats each domain in a **modular** fashion, allowing other generative user-personalization schemes to be connected easily. Finally, we provide a highly scalable inference procedure with a novel initialization schemem, allowing the model to scale to millions of users easily.

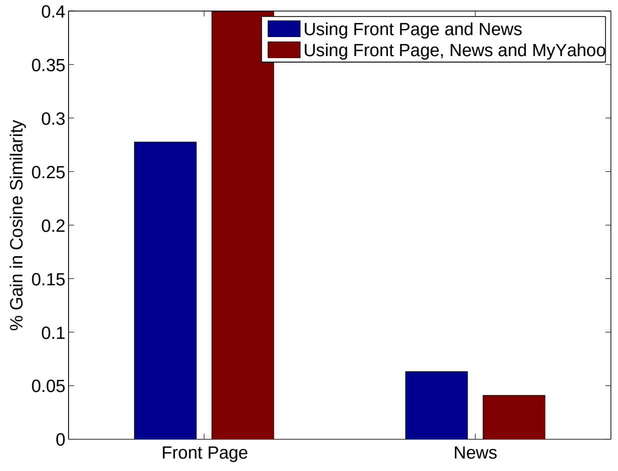

Figure 8: Comparing Frontpage and News holdout accuracy for the two domain model and the three domain model

10. REFERENCES

[1] J. Abernethy, F. Bach, T. Evgeniou, and J.-P. Vert. A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826, March 2009.

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In J.F. Elder, F. Fogelman-Soulié, P.A. Flach, and M.J. Zaki, editors, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 19–28. ACM, 2009.

[3] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proc. Intl. Conf. Machine Learning*, pages 65–72, New York, NY, 2004. ACM Press.

[4] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.

[5] K. R. Canini, L. Shi, and T. L. Griffiths. Online inference of topics with latent dirichlet allocation. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

[6] T. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. In *NIPS*, 2005.

[7] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

[8] Y. Koren, R.M. Bell, and C. Volinsky. Matrix

factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[9] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(3):503–528, 1989.

[10] John M. Mellor-Crummey and Michael L. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. *ACM Transactions on Computer Systems*, 9(1):21–65, February 1991.

[11] D. M. Mimno and A. McCallum. Topic models conditioned on arbitrary features with dirichlet-multinomial regression. In D. A. McAllester and P. Myllymäki, editors, *UAI, Proceedings of the 24th Conference in Uncertainty in Artificial Intelligence*, pages 411–418. AUAI Press, 2008.

[12] D.M. Mimno, W. Li, and A. McCallum. Mixtures of hierarchical topics with pachinko allocation. In Z. Ghahramani, editor, *International Conference on Machine Learning*, volume 227, pages 633–640. ACM, 2007.

[13] Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, and Dennis DeCoste. Naïve filterbots for robust cold-start recommendations. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *KDD*, pages 699–705. ACM, 2006.

[14] J. Pearl. *Causality: Models, Reasoning and Inference.* Cambridge University Press, 2001.

[15] C. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, second edition, 2004.

[16] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In W.W. Cohen, A. McCallum, and S.T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki*, volume 307, pages 880–887. ACM, 2008.

[17] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In Micheline Beaulieu, Ricardo Baeza-Yates, Sung Hyon Myaeng, and Kalervo Järvelin, editors, *Proceedings of the 25th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, New York, August 11–15 2002. ACM Press.

[18] A.J. Smola and S. Narayanamurthy. An architecture for parallel topic models. In *Very Large Databases (VLDB)*, 2010.

[19] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Cofi rank - maximum margin matrix factorization for collaborative ranking. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20.* MIT Press, Cambridge, MA, 2008.

[20] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In *KDD'09*, 2009.