Dionysius: A Framework for Modeling Hierarchical User Interactions in Recommender Systems

Jian Wang, Kaushik Rangadurai, David Hardtke, Krishnaram Kenthapadi
LinkedIn Corp
Sunnyvale, CA, USA
{jianwang, krangadurai, dhardtke, kkenthapadi}@linkedin.com

ABSTRACT

Real-time large-scale personalized recommendation systems power several user-facing products at many social media and web platforms. To meet business requirements, such applications must score millions of structured candidate documents associated with each query, offer a high degree of data freshness, and respond with low latency. To address these challenges, many such systems in practice make use of content based recommendation models based on logistic regression. A fundamental problem with content based models is that they are based primarily on the explicit user context in the form of user profile, but do not take into account implicit user context in the form of user interactions.

We address the following problem: *How do we incorporate user item interaction signals as part of the relevance model in a large-scale personalized recommendation system such that, (1) the ability to interpret the model and explain the recommendations is retained, and (2) the existing infrastructure designed for the (user profile) content-based model can be leveraged?* We propose *Dionysius*, a hierarchical graphical model based framework for incorporating user interactions into recommender systems. We learn a hidden fields vector for each user by considering the hierarchy of interaction signals, and replace the user profile based vector with this learned vector, thereby not expanding the feature space at all. Thus, our framework allows the use of existing recommendation infrastructure that supports content based features. We have implemented and deployed this system as part of the recommendation platform in a large professional social network. We validate the efficacy of our approach through extensive offline experiments with different model choices, as well as online A/B testing experiments. Our deployment of this system as part of the job recommendation engine has resulted in significant improvement in the quality of the retrieved results, thereby generating improved user experience and positive impact for millions of users.

1. INTRODUCTION

Personalized recommendation systems form the backbone of several user-facing products at many large Internet companies. Such systems pose unique challenges, wherein personalized recommendations need to computed in real-time from millions of structured candidate items while providing a high degree of data freshness. To address these challenges, many such systems in practice make use of content based recommendation models based on logistic regression. Content based models can be implemented by incorporating content related features across structured user fields and/or structured item fields as part of the model, in which the feature space is usually static and limited. Consider the job recommendation domain for example. In this application, structured user fields could include the user's current job title, seniority, skills, work experience, education experience, and so on. Structured job fields could include the job title, function, industry, company, seniority, skills, description, and so on. Content related features could be defined in terms of similarity between different combinations of user fields and job fields. Such features are easy to compute in both the offline training pipeline and the online prediction stage. Another benefit is that the resulting model can be easily interpreted, and explanations could be provided for the item recommendations.

A fundamental problem with content based models is that they are based primarily on the explicit user context (say, in the form of user profile), but do not take into account implicit user context in the form of user interactions. However, two users with near identical profiles may have very different preferences for job recommendations: the first user may prefer jobs with similar titles as her current position from a few selected companies, while the second user may prefer jobs with identical title as her current position but from a wider range of companies. A potential solution is to introduce dynamic user interaction signals (e.g., user m applied to job k) in the logistic regression model. A common practice is to introduce ID-level regression coefficients in addition to the global regression model in a Generalized Linear Model (GLM) setting, which is referred to as generalized linear mixed model (GLMix) [19] in the statistics literature. The key challenge for such GLMix model is the high computational complexity by significantly expanding the feature space. Imagine the scenario that we have 100M+ users and each user has 1000 non-zero coefficients on job features, this approach introduces more than 10^{11} features to learn in the model. In practice, there are various approaches to tackle this challenge, by applying dimension reduction methods (such as feature hashing) or applying parallelized block coordinate descent under the Bulk Synchronous Parallel paradigm. While the former one reduces the explanation power of the model, the latter one requires significant infrastructure change.

We address the following problem: *How do we incorporate user item interaction signals as part of the relevance model in a large-scale personalized recommendation system such that, (1) the ability to interpret the model and explain the recommendations is retained, and (2) the existing infrastructure designed for the (user profile)*

content-based model can be leveraged? We propose *Dionysius*, a hierarchical graphical model based framework for incorporating user interactions into recommender systems. We learn a hidden fields vector for each user by considering the hierarchy of interaction signals, and replace the user profile based vector with this learned vector, thereby not expanding the feature space at all. Thus, our framework allows the use of existing recommendation infrastructure that supports content based features. The intuition underlying our model is that there is a hierarchy in the strength of user interactions: in the job recommendation application as an example, explicit feedback $>$ job application $>$ job view. Our model uses the user's original profile based fields as the prior, and incorporates the interactions in a hierarchical fashion, wherein the previous layer can be thought of as the prior for the next layer. The influence of the prior information decreases as the number of behavioral observations increases. In addition, the regression coefficients are learned jointly with the user's hidden feature vector. As for prediction, the online recommendation system replaces the user's original profile based fields with the user interaction fields that are inferred by the model, thereby requiring very limited change to the underlying recommendation infrastructure.

While our proposed framework is general and applicable to any recommendation domain, we have implemented and deployed this system as part of the job recommendation platform in a large professional social network. We validate the efficacy of our approach through extensive offline experiments with different model choices and user segments, and also using online A/B testing experiments. Our deployment of this system as part of the job recommendation engine has resulted in significant improvement in the quality of the retrieved results (in terms of metrics such as VPI (views per impression) and API (applications per impression)), thereby generating improved user experience and positive impact for millions of users.

2. RELATED WORK

Context-aware recommendation systems: The notion of contextual information has been extensively studied in varied disciplines such as psychology and computer science. Bazire and Brezillon [3] present and examine 150 different definitions of context from different fields. This is not surprising, given the complexity and the multifaceted nature of the concept of context. Dourish [6] distinguishes between representational and interactional views of context. The former view considers context as static, and describes it in terms of a set of observable attributes that are known a priori. The latter view treats context as dynamic, and assumes a cyclical relationship between context and activity, where the activity gives rise to context and the context influences activity. There has been extensive work on defining and incorporating context in recommendation systems [1, 5, 22, 27, 2, 10, 16, 30, 31, 21]. There are two broad approaches to recommendation systems: content-based filtering and collaborative filtering. Content-based filtering [20, 25] assumes that descriptive features of an item indicate a user's preferences. Thus, a recommender system makes a decision for a user based on the descriptive features of other items the user likes or dislikes. Usually, the system recommends items that are similar to what the user liked before. Collaborative filtering [11, 26, 12, 18, 29, 24, 8] on the other hand assumes that users with similar tastes on some items may also have similar preferences on other items. In contrast, we propose a hierarchical graphical model to incorporate user interactions into the recommendation model. Our work can be viewed as bringing together the representational (user profile) and interactional (user interactions) views of context, especially in the job recommendation setting.

Job Recommendation and Professional Social Networks: There is extensive work on how people find jobs, how they make use of their connections to obtain jobs, and how to match people and jobs, spanning diverse areas such as organizational psychology and computer science [9, 17, 23, 14, 15, 28]. While it is desirable to take into account the person-environment (P-E) fit (subdivided further into person-organization (P-O) fit, person-vocation or occupation (P-V) fit, and person-group (P-G) or person-team fit) in addition to person-job (P-J) fit [7, 13], we focus primarily on the person-job fit in this paper.

3. PROBLEM SETTING

Before formally describing the hierarchical user interaction model that we propose, we first briefly review the basics of existing recommender system that is built with logistic regression, which is a common practice in industry due to its decent performance with low computational complexity.

Logistic regression is widely used to solve two-class classification problems with decent performance, e.g. predicting the probability of a user m having the observed action $y_{m,k} = 1$ on item k. We denote the feature vector between user m and item k as $\mathbf{x_{m,k}}$ and the coefficient vector as β. The probability could be estimated based on all available features as follows:

$$p(y_{m,k} = 1) = \frac{1}{1 + exp\{-\beta^{\mathbf{T}}\mathbf{x_{m,k}}\}}$$

Assuming that the prior distribution of each model parameter is a Gaussian centered on zero, the optimal coefficient vector β can be learned from the training data using the maximum a posteriori probability (MAP) estimation.

The key of building the logistic regression model with good performance is to introduce valuable features. Content-based models can be implemented by introducing content-related features to the model, including features from user profile-based fields, job profile-based fields, user-job similarity features in content and so on. Generally speaking, the content-related feature space is pretty static and limited. For example, user profile-based fields include working experience, education experience, skills and so on. Job profile-based fields include job title, function, industry, company, seniority, description and so on. User-job similarity features could be similarity between any combination of user profile field or job profile field. In addition, these features are easy to compute in both the offline training pipeline and the online prediction stage.

A fundamental problem with content-based models is that they are based primarily on the member's profile. However, two members with near identical profiles may have very different preferences for job recommendations: the first member may prefer jobs with similar titles as her current position from a few selected companies, while the second member may prefer jobs with identical title as her current position but from a wider range of companies. The natural solution is to introduce dynamic user interaction signals in logistic regression model, which simulates the idea of collaborative filtering. For example, if user m applied to job k, we should incorporate this signal into the model. A common practice is to introduce ID-level regression coefficients in addition to the global regression model in a Generalized Linear Model (GLM) setting, which is referred to as generalized linear mixed model (GLMix) [19] in the statistics literature. In our case, we can introduce outer product between user index m and job features, as well as job index k and user features. The intuition is that these ID-level features could capture job fields that are the best match for user m and user fields that are the best match for job k.

$$p(y_{m,k}=1) = \frac{1}{1+exp\{-\beta^{\mathbf{T}}\mathbf{x_{m,k}} + \alpha_{\mathbf{m}}{}^{T} j_k + \gamma_{\mathbf{j}}{}^{T} u_m\}}$$

where β is the coefficient vector of the global regression model, and $\alpha_{\mathbf{m}}$ and $\gamma_{\mathbf{j}}$ are the coefficient vectors specific to user m and job k respectively.

The key challenge for such GLMix model is the high computational complexity by significantly expanding the feature space. Imagine the scenario that we have 100M+ users and each user has 1000 non-zero coefficients on job features, this approach introduces more than 10^{11} number of features to learn in the model. In practice, there are various approaches to tackle this challenge, by applying dimension reduction methods (such as feature hashing) or applying parallelized block coordinate descent under the Bulk Synchronous Parallel paradigm. While the former one reduces the ease of interpretability of the model, the latter one requires significant change in the recommendation infrastructure.

We address the following problem:

How do we incorporate user item interaction signals as part of the relevance model in a large-scale personalized recommendation system such that, (1) the ability to interpret the model and explain the recommendations is retained, and (2) the existing infrastructure designed for the (user profile) content-based model can be leveraged?

4. DIONYSIUS: A FRAMEWORK FOR MODELING HIERARCHICAL USER INTERACTIONS

We next present the desiderata for addressing our problem and the notations/preliminaries needed to describe our framework. Then, we describe our hierarchical user interaction model, followed by a brief overview of how we implement and deploy as part of the large-scale recommendation system.

4.1 Desirable Properties

We first highlight the desirable properties of a framework for modeling user item interactions in our recommendation setting:

Interpretability and explainability of recommendations: The model should be easy to interpret, and the recommendations easy to explain.

Differential weights for different action types: Stronger interaction types (such as applications) should be given greater weight than weaker ones (such as views).

Graceful fallback: The model should be able to handle users with significant interaction activity as well as users with no interaction activity. In other words, it should gracefully fallback to the user profile based setting when there is no interaction activity.

Infrastructure and deployment considerations: The proposed approach should not require significant change to the existing recommendation infrastructure, which is designed as a content based recommendation system.

4.2 Preliminaries

In this paper, the key problem is to predict the probability for a user m applying to a job k. With that information, the system can rank all potential jobs according to the probability and recommend top ones to user m. The following notations are used in the paper.

- $m = 1, 2, ..., M$: the index of the user.
- $k = 1, 2, ..., N$: the index of the job.
- $y_{v,m,k}$: the binary label that indicates the user's viewing behavior for job k. If $y_{v,m,k} = 1$, it indicates that the user m clicks to view job k. Otherwise $y_{v,m,k} = -1$.
- $y_{a,m,k}$: the binary label that indicates the user's application behavior for job k. If $y_{a,m,k} = 1$, it indicates that the user m clicks to apply to job k. Otherwise $y_{a,m,k} = -1$.
- $\mathbf{u_{p,m}}$: the vector of profile-based fields associated with user m. The vector includes static demographic features that are derived from the user profile information.
- $\mathbf{u_{v,m}}$: the vector of view-based fields associated with user m. The vector is tuned according to the user's viewing behavior.
- $\mathbf{u_{a,m}}$: the vector of application-based fields associated with user m. The vector is tuned according to the user's application behavior.
- $\mathbf{j_k}$: the vector of fields associated with job k. The vector includes static features that are derived from the job description.
- $\mathbf{x_{m,k}}$: the feature vector that is associated with the user m profile and job k. It might include user profile-based fields $\mathbf{u_{p,m}}$, job profile-based fields $\mathbf{j_k}$, and similarity-based features between user profile-based fields $\mathbf{u_{p,m}}$ and job profile-based fields $\mathbf{j_k}$.
- $D = \{D_1, ..., D_m, ..., D_M\}$: The observed data of all users.
- $D_m = \{y_{v,m,k}, y_{a,m,k}, \mathbf{u_{p,m}}, \mathbf{j_k}\}$: A set of observed data associated with user m. Each observation is associated with four parts: the user viewing behavior, the user application behavior, the user profile-based fields, and the job fields .
- β_v: the d-dimensional vector of regression coefficients to predict the user's viewing behavior $y_{v,m,k}$.
- β_a: the d-dimensional vector of regression coefficients to predict the user's application behavior $y_{a,m,k}$.
- σ_v: variance of the user view-based feature vector $\mathbf{u_{v,m}}$.
- σ_a: variance of the user application-based feature vector $\mathbf{u_{a,m}}$.

4.3 Description of Hierarchical User Interaction Model

We propose a hierarchical graphical model to incorporate user interactions into job recommendations. Instead of introducing ID-level regression coefficients, we learn the user hidden feature vector by considering their interaction signals (job views, applications). In the global regression model, we replace the user profile-based vector with the user hidden feature vector, thereby not expanding the feature space at all. It allows the use of existing infrastructure that supports content-based features. The user hidden feature vector is learned based on their job viewing and applying behavior as well as other explicit positive/negative feedbacks on the recommended jobs. The intuition underlying our model is that there is a hierarchy in the strength of user activity: job application > job view. Note that we describe our model with two types of user behaviors while it is straight-forward to add other types of interactions (such as saving a job) in the model as an additional layer in the hierarchy.

Let's go through the model by each component.

Model parameters . The first component specifies the parameter likelihood in the model. In particular, β_v is the coefficient vector for regression model which predicts the user viewing behavior. β_a is the coefficient vector for regression model which predicts the user applying behavior. σ_v controls the variance of the distribution where user view-based fields $u_{v,m}$ is drawn from. σ_a controls the variance of the distribution where user application-based fields $u_{a,m}$ is drawn from.

We assume that parameters are sampled from a Gaussian distribution. In details,

$$\begin{aligned}\beta_v &\sim N(\mu_{\beta_v}, \sigma_{\beta_v}) \\ \beta_a &\sim N(\mu_{\beta_a}, \sigma_{\beta_a}) \\ \sigma_v &\sim N(\mu_{\sigma_v}, \sigma_{\sigma_v}) \\ \sigma_a &\sim N(\mu_{\sigma_a}, \sigma_{\sigma_a})\end{aligned} \tag{1}$$

For easier interpretation, we denote
$\phi = (\mu_{\beta_v}, \sigma_{\beta_v}, \mu_{\beta_a}, \sigma_{\beta_a}, \mu_{\sigma_v}, \sigma_{\sigma_v}, \mu_{\sigma_a}, \sigma_{\sigma_a})$

User feature vector . The second component specifies the likelihood of the user feature vector. $u_{p,m}$, i.e., the user profile-based fields, is extracted from the user static profile. We assume that $u_{v,m}$, i.e., the user view-based fields, follows the Gaussian distribution with $u_{p,m}$ as mean and σ_v as variance. Similarly, we assume that $u_{a,m}$, i.e., the user application-based fields, follows the Gaussian distribution with $u_{v,m}$ as mean and σ_a as variance.

$$\begin{aligned}\mathbf{u_{v,m}} &\sim N(\mathbf{u_{p,m}}, \sigma_\mathbf{v}) \\ \mathbf{u_{a,m}} &\sim N(\mathbf{u_{v,m}}, \sigma_\mathbf{a})\end{aligned} \tag{2}$$

In practice, σ_v and σ_a could be tuned manually to control the weight of the prior fields that come from the user profile. The higher the variance σ_v, the less important the user profile fields $\mathbf{u_{p,m}}$ are. It indicates that the model gives more weight to the user viewing behavior and cares less about the user original profile. Similarly as for the user application behavior.

User interaction signal . The third component specifies the likelihood of the user interaction signal. In the former one, the user viewing behavior $y_{v,m,k}$ is dependent on the user view-based vector $u_{v,m}$ and the regression model β_v. In the latter one, the user application behavior $y_{a,m,k}$ is dependent on the user application based vector $u_{a,m}$ and the regression model β_a. We use logistic regression as the core model to predict user action.

$$p(y_{v,m,k}|\mathbf{u_{v,m}}, \beta_v) = \frac{1}{1 + exp(-y_{v,m,k}({\beta_v}^T f(j_k, \mathbf{u_{v,m}})))} \tag{3}$$

$$p(y_{a,m,k}|\mathbf{u_{a,m}}, \beta_a) = \frac{1}{1 + exp(-y_{a,m,k}({\beta_a}^T f(j_k, \mathbf{u_{a,m}})))}$$

Assuming that the data is independently and identically distributed, we can present the data likelihood as:

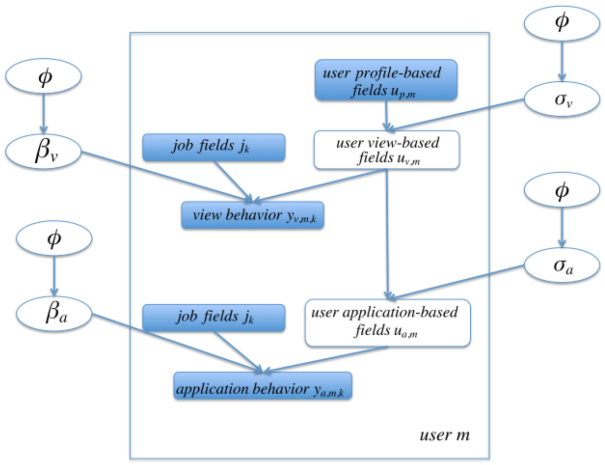

Figure 1: Illustration of dependencies of variables in the hierarchical user interaction model. It shows the observed dataset of user m, i.e., $D_m = \{y_{v,m,k}, y_{a,m,k}, \mathbf{u_{p,m}}, \mathbf{j_k}\}$. All variables in shades are observed ones while others are hidden variables in the model. As shown in the plot, the user view-based fields $u_{v,m}$ is drawn from distribution of user profile-based fields $u_{p,m}$ and variance σ_v. Similarly, the user application-based fields $u_{a,m}$ is drawn from distribution of user view-based fields $u_{v,m}$ and variance σ_a. The user view behavior $y_{v,m,k}$ is dependent on the regression model β_v, the job fields j_k and the user view-based fields $u_{v,m}$. Similarly, the user application behavior $y_{v,m,k}$ is dependent on the regression model β_a, the job fields j_k and the user view-based fields $u_{a,m}$.

$$\begin{aligned}p(D|\phi) &= \int p(D, \theta_g|\phi) d\theta_g \\ &= \int p(D|\theta_g, \phi) p(\theta_g|\phi) d\theta_g \\ &= \int [\prod_{m=1}^{M} p(y_m|\theta_g, \phi)] p(\theta_g|\phi) d\theta_g\end{aligned} \tag{4}$$

where $\theta_g = (\beta_a, \beta_v, \sigma_a, \sigma_v)$ is a random variable denoting the joint distribution of the global random variables.

In particular, the data likelihood for user m can be presented as

$$p(y_m|\theta_g, \phi) = \int p(y_m|\theta_m, \theta_g, \phi) p(\theta_m|\theta_g, \phi) d\theta_m, \tag{5}$$

$$\begin{aligned}p(y_m|\theta_m, \theta_g, \phi) p(\theta_m|\theta_g, \phi) = \prod_{k=1}^{K_m} [p(y_{v,m,k}|\mathbf{u_{v,m}}, \beta_v) p(y_{a,m,k}|\mathbf{u_{a,m}}, \beta_a)] \\ p(u_{a,m}|u_{v,m}, \sigma_a) p(u_{v,m}|\sigma_v)\end{aligned} \tag{6}$$

where $\theta_m = (\mathbf{u_{v,m}}, \mathbf{u_{a,m}})$ is a random variable denoting the joint distribution of the view based vector and application based vector random variables for each user m.

Maximizing the likelihood of $p(D|\phi)$ is equivalent to maximizing the log likelihood, $L(D|\phi) = \ln p(D|\phi)$.

There is no known closed-form solution for the estimation of model parameters. We follow the Bayesian method [4] to derive an Expectation-Maximization (EM) based iterative process to find an approximate solution. In the E step, we fix the regression model and tune the user interaction-based vector. In the M step, we fix the user interaction-based vector and learn the regression model accordingly. The iterative process converges when there is minimal change in regression model coefficients and user interaction-based vectors.

Thus, our proposed approach can be summarized as follows:

- We propose a hierarchical graphical model to incorporate user interactions into the recommendation model.
- The user hidden feature vector is learned based on the user's job viewing behavior and job applying behavior.
- Three layers of user information would be leveraged hierarchically in the recommendation stage, including the user's profile, user's viewing behavior, and the user's applying behavior. The previous layer would act as the prior for the next layer. The influence of the prior information decreases as the number of behavior observations increases.
- The regression coefficients are learned jointly with the user's hidden feature vector.

4.4 Implementation and Deployment as part of the Recommendation System

We next briefly describe the overall design and architecture of the recommendation system deployed at a large professional social network, focusing on how we implement and deploy our framework for modeling hierarchical user item interactions as part of the existing recommendation system infrastructure.

4.4.1 Offline System for Generating and Incorporating User Item Interaction Model

In the offline system, the interaction based fields vectors for users are generated and pushed to an interaction store (a distributed key value store) through the following steps:

1. Train the hierarchical user interaction model using the user's interactions and store the regression coefficients for prediction.
2. Use the trained user interaction model and the user's interactions in the last N days to predict the user's interaction based hidden vector (this is updated periodically).
3. Push the user's interaction based fields vector to a distributed key value store so that it can be retrieved by the online system.

4.4.2 Online Query Processing and Recommendation System

In the online component, the following steps take place after a call to the recommendation service is made for a user:

1. Use the user id to fetch user's interaction based fields vector from the interaction store (the data is pushed to this store from the offline system).
2. If this hidden vector is missing (e.g., for users with no interactions), substitute with the (content-based) user fields vector from the member store.

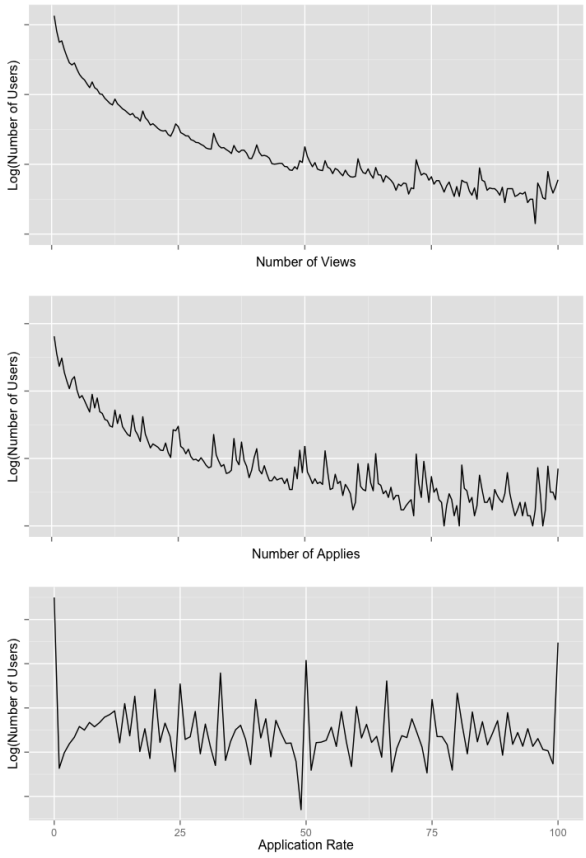

Figure 2: Histogram of user interaction in the dataset. In all three plots, y axis corresponds to Log(number of users). x axis in three plots corresponds to number of views, number of applies and application rate. Exact number is omitted for business concern.

3. Form the query based on the user fields, and query against an inverted index system (e.g., Lucene) of items to retrieve a candidate set of items to score.
4. Score the candidates using a machine learned model (e.g., based on logistic regression) and return the ranked list of items.

5. EXPERIMENTS

5.1 Research Questions

As described before, the hierarchical user interaction model could be applied to any domain. Without loss of generality, we evaluate it in the job recommendation domain here. In the experiment, We consider two popular user interactions, including job views and job

Table 1: Models to compare

Model	User interaction signal	hierarchy or not
M-baseline	None	None
M-view	job views	None
M-apply	job applications	None
M-viewApply	job views, job applications	Hierarchical model

applications. In the professional social network, summaries of job vacancy postings are shown to the user. We call this an **impression** (each job vacancy summary shown to the user is considered to be impressed). Job seekers can open a new page to view the entire job description, which we call as a **view** action. In addition, users will apply to the job if they wish to be considered for the position, which we call as an **apply** action.

We first list all research questions, followed by the experimental setup, data analysis and performance analysis. To show the effectiveness of the hierarchical user interaction model, we design our experiments to answer the following research questions:

- Is it helpful to consider signals from user interactions in the model? How much lift do these signals have in terms of the overall recommendation performance?
- Is it helpful to incorporate different types of user interaction signals in a hierarchical way?
- How is the performance comparison for different user segments?

5.2 Evaluation Metrics

We used both offline and online measures to quantify the effectiveness of the hierarchical user interaction models.

In the offline evaluation, we compute the area under receiver operating characteristic curve (ROC AUC), which represents the quality of the item recommendation system (viewed as a binary classifier). A (user, job) pair is labeled as a positive example if the user applies to the job during the evaluation period, and as a negative example otherwise. The ROC curve is obtained by plotting the true positive rate (recall) against the false positive rate at various choices of the threshold in the recommendation model. ROC AUC can be interpreted as the probability that the binary classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance.

In the online evaluation, we use the following two business metrics: (1) the job view rate (VPI), which is defined as the ratio of the number of jobs viewed by users to the number of job impressions (number of jobs presented to users); (2) the job application rate (API), which is defined as the ratio of the number of jobs applied to by users to the number of job impressions. Higher job view and application rates correspond to more relevant jobs being shown to users, and hence are desirable. We also present the number of impressions, the number of views, and the number of applications. For all of these metrics, we report only the relative change between two models, and omit the absolute numbers due to business concerns.

5.3 Models to Compare

In our experiments, we compare the basic regression model, as well as the hierarchical user interaction model with different components. As shown in Table 1, **M-baseline** is the basic regression model with no user interaction signals. **M-view** is the user interaction model with viewing behavior. **M-apply** is the user interaction model with application behavior. **M-viewApply** is the hierarchical user interaction model with both user viewing behavior and application behavior.

All models use the same set of features for the core regression model. Features include those that are extracted from the user profile-based fields (current title, current industry, and so on) and those extracted from the job fields (job title, company, skills and so on), as well as similarity features between user fields and job fields. Extensive feature engineering could be explored to improve the model performance, which is beyond the scope of this paper.

5.4 Dataset

As mentioned earlier in the paper, we apply our model in the context of a job recommender system and use a real-world dataset from a professional social network to evaluate our models.

5.4.1 Training Stage

In the model training stage, we used two sets of dataset, one for tuning user interaction-based fields and another one for training the logistic regression model.

Training: dataset for tuning user interaction-based fields We first collect the data to tune user interaction-based fields if a user viewed a job or applied to a job from 2016/01/01 to 2016/01/09. This sample data contains millions of users that applied to a reasonable number of jobs in that period [1]. Histograms of user interactions are demonstrated in Figure . As presented in the figure, the distribution of both user views and applications follow the power law distribution, which indicates that a few users who applied to or viewed a lot jobs while majority of users who applied to/view only a few jobs. In Figure , we also show the histogram of application rate, i.e., # of applications divided by # of views. There are two peaks in the plot, which corresponds to users who didn't apply to any jobs that they viewed and users who applied to almost all jobs that they viewed. In the performance analysis, we evaluate the performance of models on these different user segments.

Training: dataset for training logistic regression model We then collect the dataset to learn the logistic regression model with the user interaction-based fields. if a user applied to a job from 2016/01/10 to 2016/01/15, we collect the data as a positive label. If a user viewed or applied to job at position k, we collect all recommendations from position 0 to position $k-1$ as negative label. In addition, we random sample from all jobs that have been applied on a particular day as random negative data to simulate the real-world scenario. For example, if user m applied to jobs on 2016/01/11, we collected all jobs that have been applied by other users on that day and removed jobs that were applied by user m. We random sampled from this set as random negative labels for user m.

5.4.2 Testing Stage

In the model testing stage, we used two sets of dataset, one for learning user interaction fields and another one for making recommendations.

Testing: dataset for learning user interaction fields We first collect user interactions from 2016/01/16 to 2016/01/25 to learn the user interaction-based fields. During the performance analysis, we segment users into different groups based on their interaction in this time period. In details, users are put into five segments, *high APP high APV, high APP low APV, zero APP high View, zero APP low View, zero APP and zero View*. *high APP* corresponds to users with at least 1 application while *zero APP* corresponds to users with no applications. *high APV* corresponds to users with higher application per view rate while *low APV* corresponds to users with lower application per view rate. *high View* corresponds to users with higher number of views. *low View* corresponds to users with lower number of views while *zero View* corresponds to users with no views. Higher application per view rate is chosen based on 75% percentile.

[1]Exact number is omitted for business concern

Testing: dataset for making recommendations We collect user application data from 2016/01/26 to 2016/01/30. The positive data and negative data are labeled similarly as we did in the training stage. All user interaction-based fields that are learnt in the last step were used here in the recommendation stage, along with the regression model that is learnt in the training stage. Note that if the user doesn't have any interaction in the last step, user interaction-based fields would fall back as his profile-based fields.

5.5 Offline Evaluation

We present the performance of different models in Table 3.

5.5.1 User interaction model VS Non-user interaction model

The very basic question is whether the user interaction model does provide better performance by considering signals from the users' interactions. It is not surprising that we do see significantly better performance from models that leverage user interactions, compared to the baseline model with no interactions **M-baseline**. In addition, we observe a better overall performance of model **M-apply**, compared with **M-view**. It indicates that the users past application behavior gives more reliable signals of users job-seeking intention in the future.

In Table , we show a real example of user profile, his interactions and recommendations from each model. As shown in the table, this user currently works as *Business Analyst* and viewed jobs with title *Software Engineer, Product Manager, Business Analyst*. Some jobs that he viewed match with his profile information yet not all of them. Among all jobs that he viewed, he chose the apply to jobs with title *Product Manager, Business Analyst*. Based on this information, the baseline model **M-baseline** recommends jobs with title similar to *Business Analyst* that match with his profile information. It didn't consider the member's job seeking intention in *Software Engineer* or *Product Manager*. On the other hand, the user interaction model successfully incorporated the user interaction signals in the recommendations. **M-view** recommends jobs that are consistent with his viewing behavior while **M-apply** recommends jobs that are consistent with his application behavior. The hierarchical model **M-viewApply** considers signals from both types of interactions and make recommendations. Note that the actual recommendations from **M-view** and **M-viewApply** are different although they share the same job title. Job details are not shown here due to privacy concerns.

5.5.2 Active users VS Passive users

When we compare the performance of all user interaction models for all the users, the **M-viewApply** performs significantly better than the **M-baseline**. However, in order to truly understand each model's prediction power, we analyze their performance on four user segments, *high APP high APV, high APP low APV, zero APP high APV, and zero APP low APV*.

In the first segment for users with *high APP high APV*, it corresponds to users with lots of job views and job applications. All user interaction models perform significantly better than the baseline model. In addition, **M-viewApply** performs better than **M-apply** which is better than **M-view**. It demonstrates the effect of using a hierarchical model **M-viewApply** that incorporates different types of user interactions in making recommendations.

In the second segment for users with *high APP low APV*, it corresponds to users with lots of job views and low number of applications. In this case, **M-view** performs better than **M-apply** since it incorporates more signals from the user viewing history. Interestingly we observe a better performance of **M-viewApply**, compared with **M-view**. The difference is triggered by the different regression model that are learnt in these two models.

In the third and fourth segment for users with *zero App* and at least 1 views, it corresponds to users with at least 1 job views and no applications. On the one hand, we observe same performance of **M-apply** and **M-baseline** since no user application signal has been incorporated in the **M-apply** model. On the other hand, **M-view** is slightly better than **M-baseline** while **M-view-apply** is significantly better than the baseline.

In the last segment for users with *zero App zero View*, it corresponds to users with no interactions in the prediction stage. We observe significantly better performance from all user interaction models, compared with the baseline model **M-baseline**. It indicates that the regression model learnt in the user interaction model has better prediction power than the baseline model, which is trained without considering user interactions.

5.6 Online Evaluation

5.6.1 Online A/B Testing Setup

In the online A/B testing, we evaluate the performance of recommender systems with real-world users on a professional social network. We randomly select 5% users that visit the site for each model and present the corresponding recommendations to each user group. The difference of the performance between two user buckets are reported in the performance analysis. A significance level of 0.05 with the paired two-tailed t-test is used to compare two models. We let each model to run for one week to burn in the novelty effect (active job seekers tend to apply to any new jobs they see due to a change in the recommendation model) and report the comparison of the performance of the models in the subsequent time period.

5.6.2 Results from Online Experiments

We observed the following results in the online A/B testing, comparing the model **M-viewApply** against the baseline: **API: +3.6%** and **VPI: +3.5%**. Further, we presented 4.1% more impressions, with 7.7% more views and 7.8% more applications. Not surprisingly, it is clear that the model **M-viewApply** performs significantly better than the baseline by incorporating user interaction signals. All key metrics, including # of applications and # of views are improved around 7% while the corresponding API and VPI are improved by around 3%. It indicates that the model **M-viewApply** learns the user's true job seeking intention from their previous interaction signals, which might be different by the user profile.

6. CONCLUSION AND FUTURE WORK

We proposed *Dionysius*, a hierarchical graphical model based framework for incorporating user interactions into recommender systems. Our framework enables the incorporation of user item interaction signals as part of the relevance model in a large-scale personalized recommendation system, with minimal change to the existing recommendation infrastructure, while retaining the ability to interpret the model and explain the recommendations. As part of our proposed model, we learned a hidden fields vector for each user by considering the hierarchy of interaction signals, and replaced the user profile based vector with this learned vector, thereby not expanding the feature space at all. Our implementation and deployment of this system as part of the job recommendation platform in a large professional social network demonstrated the efficacy and practicality of our framework, and has also resulted in significant

Table 2: Example of user profile, his activities and recommendations from each model. For privacy concerns, only the job title has been shown in the table. Note that jobs with the same title might refer to different unique jobs.

User Profile Title	Business Analyst
Jobs that are viewed by the user	Software Engineer, Product Manager, Business Analyst
Jobs that are applied by the user	Product Manager, Business Analyst

M-baseline	Business Analyst, Consultant - Business Intelligence and Business Analytics, Data Analyst
M-view	Business Analyst, Product Manager, Software Engineer
M-apply	Business Analyst, Product Manager, Product Manager
M-viewApply	Business Analyst, Product Manager, Software Engineer

Table 3: Offline ROC AUC analysis on different models and user segments. Users are segmented into five segments, *high APP high APV, high APP low APV, zero APP high View, zero APP low View, zero APP and zero View*. *high APP* corresponds to users with at least 1 application while *zero APP* corresponds to users with no applications. *high APV* corresponds to users with higher application per view rate while *low APV* corresponds to users with lower application per view rate. *high View* corresponds to users with higher number of views. *low View* corresponds to users with lower number of views while *zero View* corresponds to users with no views.

	all users	high App, high APV	high App, low APV	zero App, high View	zero App, low View	zero App, zero View
M-baseline	0.612	0.612	0.571	0.602	0.604	0.618
M-view	0.638 (+4.2%)	0.663	0.604	0.606	0.619	0.669
M-apply	0.643 (+5.1%)	0.673	0.603	0.603	0.605	0.675
M-viewApply	**0.644 (+5.2%)**	**0.682**	**0.606**	**0.612**	**0.623**	**0.677**

improvement in the quality of the recommendation results for millions of users.

In future, we plan to extend and apply this framework for the items, that is, incorporate the signals from users that interacted (e.g., viewed/applied to a job) with the item to enhance the representation of the item in a hierarchical fashion, wherein the strength of the interaction is factored in. Another fruitful direction to pursue is to apply our framework for associations beyond interactions based on relationship in the structured fields. For example, in the job recommendation application, we can take into account jobs that have the same title as the user, jobs that have the same title and skills as the user, and so on, and apply the hierarchical model to enhance the representation for new users or those with no interaction activity. Likewise, we can apply such association to enhance the representation for new jobs or those with views or applications from users.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*. 2011.

[2] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Context-aware Recommender Systems Workshop at RecSys09*, 2009.

[3] M. Bazire and P. Brézillon. Understanding context before using it. In *International Conference on Modeling and Using Context*. 2005.

[4] M. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, University of London, 2003.

[5] L. Do and H. W. Lauw. Modeling contextual agreement in preferences. In *WWW*, 2014.

[6] P. Dourish. What we talk about when we talk about context. *Personal and ubiquitous computing*, 8(1), 2004.

[7] J. R. Edwards. *Person-job fit: A conceptual integration, literature review, and methodological critique.* John Wiley & Sons, 1991.

[8] N. Golbandi, Y. Koren, and R. Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *WSDM*, 2011.

[9] M. Granovetter. *Getting a job: A study of contacts and careers*. 1995.

[10] N. Hariri, B. Mobasher, and R. Burke. Query-driven context aware recommendation. In *RecSys*, 2013.

[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 1999.

[12] R. Jin, L. Si, C. Zhai, and J. Callan. Collaborative filtering with decoupled models for preferences and ratings. In *CIKM*, 2003.

[13] A. L. Kristof. Person-organization fit: An integrative review of its conceptualizations, measurement, and implications. *Personnel psychology*, 49(1), 1996.

[14] H. Liu, M. Amin, B. Yan, and A. Bhasin. Generating supplemental content information using virtual profiles. In *RecSys*, 2013.

[15] H. Liu, A. Goyal, T. Walker, and A. Bhasin. Improving the discriminative power of inferred content information using segmented virtual profile. In *RecSys*, 2014.

[16] X. Liu and K. Aberer. Soco: A social network aided context-aware recommender system. In *WWW*, 2013.

[17] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel. Matching people and jobs: A bilateral recommendation approach. In *HICSS*, 2006.

[18] B. Marlin and R. S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *ICML*, 2004.

[19] C. McCulloch and J. Neuhaus. *Generalized linear mixed models*. Wiley Online Library, 2001.

[20] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *DL*, 2000.

[21] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas. Gaussian process factorization machines for context-aware recommendations. In *SIGIR*, 2014.

[22] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *RecSys*, 2009.

[23] I. Paparrizos, B. B. Cambazoglu, and A. Gionis. Machine learned job recommendation. In *RecSys*, 2011.

[24] D. Parra-Santander and P. Brusilovsky. Improving collaborative filtering in social tagging systems for the recommendation of scientific articles. *Web Intelligence and Intelligent Agent Technology*, 1, 2010.

[25] M. Pazzani, D. Billsus, S. Michalski, and J. Wnek. Learning and revising user profiles: The identification of interesting web sites. In *Machine Learning*, 1997.

[26] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML*, 2003.

[27] K. Verbert, N. Manouselis, X. Ochoa, M. Wolpers, H. Drachsler, I. Bosnic, and E. Duval. Context-aware recommender systems for learning: A survey and future challenges. *IEEE Transactions on Learning Technologies*, 5(4), 2012.

[28] J. Wang and D. Hardtke. User latent preference model for better downside management in recommender systems. In *WWW*, 2015.

[29] J. Wang, B. Sarwar, and N. Sundaresan. Utilizing related products for post-purchase recommendation in e-commerce. In *RecSys*, 2011.

[30] J. Weston, R. J. Weiss, and H. Yee. Nonlinear latent factorization by embedding multiple user interests. In *RecSys*, 2013.

[31] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, 2014.