Personalized Recommendation on Dynamic Content Using Predictive Bilinear Models

Wei Chu
Yahoo! Labs.
2821 Mission College Blvd
Santa Clara, CA 95054
chuwei@yahoo-inc.com

Seung-Taek Park
Yahoo! Labs.
2821 Mission College Blvd
Santa Clara, CA 95054
parkst@yahoo-inc.com

ABSTRACT

In Web-based services of dynamic content (such as news articles), recommender systems face the difficulty of timely identifying new items of high-quality and providing recommendations for new users. We propose a feature-based machine learning approach to personalized recommendation that is capable of handling the cold-start issue effectively. We maintain profiles of content of interest, in which temporal characteristics of the content, e.g. popularity and freshness, are updated in real-time manner. We also maintain profiles of users including demographic information and a summary of user activities within Yahoo! properties. Based on all features in user and content profiles, we develop predictive bilinear regression models to provide accurate personalized recommendations of new items for both existing and new users. This approach results in an offline model with light computational overhead compared with other recommender systems that require online re-training. The proposed framework is general and flexible for other personalized tasks. The superior performance of our approach is verified on a large-scale data set collected from the Today-Module on Yahoo! Front Page, with comparison against six competitive approaches.

Categories and Subject Descriptors

H.1.0 [**Models and Principles**]: General; H.3.3 [**Information Search and Retrieval**]: Information filtering; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithms, Experimentation, Design, Performance

Keywords

Personalization, Dynamic Features, Bilinear, Regression, Ranking, User and Content Profiles, Recommender Systems

1. INTRODUCTION

The Internet provides an unparalleled opportunity for organizations to deliver digital content to their visitors instantaneously. Content consumers usually have short attention span, while possibly a large number of content venders. The biggest challenge most organizations face is not lack of content, but how to optimize the content they already own by identifying the most appropriate customers at the right time. Personalized recommendation has become a desirable feature of e-business Web sites to improve customer satisfaction and customer retention [8], by tailoring content presentation to suit an individual's needs rather than take the traditional "one-size-fits-all" approach.

Personalized recommendation involves a process of gathering and storing information about site visitors, managing the content assets, analyzing current and past user interactive behavior, and, based on the analysis, delivering the right content to each visitor [31]. Search engines help index available content assets and return relevant information to users, if the users are looking for something specific that can be summarized as a keyword query. However, in many cases, users are looking for things might interest them, but do not have concrete desideration in mind when browsing a Web site. In such cases, it is a recommendation engine that presents the most plausible content that the user may want, based on her interests as demonstrated by her *past activities*. Traditional recommendation engines could be distinguished into three different approaches: rule-based filtering, content-based filtering, and collaborative filtering [32]. Rule-based filtering creates a user-specific utility function and then applies it to the items under consideration. This approach is closely related to customization, which requires users to identify themselves, configure their individual settings, and maintain their personalized environment over time [21]. It is easy to fail since the burden of responsibility falls on the users. Content-based filtering generates a profile for a user based on the content descriptions of the items previously rated by the user. The main drawback of this approach is the recommended items are similar to the items previously seen by the user. Mladenic [30] provided a survey of the commonly used text-learning techniques in the context of content filtering. Collaborative filtering (CF) is one of the most successful and widely used recommender system technology [37]. CF analyzes users' ratings to recognize commonalities between users on the basis of their historical ratings, and then generates new recommendations based on like-minded users' preferences. CF provides a good solution to "a closed world", where overlaps in ratings across users are relatively high and the universe of content items is almost static.

In many scenarios, such as news filtering [15], where the content universe changes rapidly and significant portion of users are new users, CF will suffer from the cold-start problem. Several hybrid recommender systems have been devel-

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

oped to tackle the cold-start problem by combining two or more recommendation techniques. The inability of CF to recommend new items is commonly leveraged by coupling with a content-based filtering, such as in Fab [3], a recommender system for the Web content. Burke [10] provided a comprehensive analysis of approaches to generating hybrid recommendation engines.

Although hybridization can alleviate some of the weaknesses associated with CF and other recommendation techniques, there are still a few important issues that haven't been well studied in literature:

- *Dynamic Content*: We consider not only the item set undergoes insertions and deletions frequently, but also the content value and then the appraisement from users are changing rapidly as well. For example, the lifetime of breaking news on the Internet is usually a couple of hours, and the value of the news (such as click through rate) is decaying temporally as people get to know it, see Figure 3(a) for an example. Traditional recommender systems usually treat users' feedback static, so that feedback on the same items given at different time stamps is still comparable. This assumption doesn't hold on dynamic content. Rebuilding the model on very recent data is typically an expensive task, and tends to lose long-term interests of users. On dynamic content, recommender systems always face the cold-start problem for new items.

- *Users with Open Profiles*: A typical user profile in a CF system is a list of ratings on items of interest. In practice, we can legally collect user information to develop a general profile for a site visitor [19], which is not limited to the content universe only. The general profile may include declared demographic information, activities on relevant sites, consumption history, etc. The objective is to provide valuable insight into users' preferences, interests and wants. Clearly, the general profile can help tackle the cold-start problem on new users. Demographic recommender systems, e.g. [34], aim to segment users based on personal attributes and make recommendations according to demographic classes. However, the history of user ratings and content features haven't been jointly exploited to form "people-to-people" correlation.

In this paper, we propose a machine learning approach to handling both issues in personalized recommendation. The key idea is to maintain profiles for both content and users, and build a feature-based bilinear regression model to quantify the associations between heterogeneous features by fitting the historical interactive data. The feature-based predictive model can then be applied to recommending new and existing items for both new and existing users.

The goodness of dynamic content over time is a crucial ingredient in content management. We insert dynamic features, such as instantaneous click-through rate (CTR) to indicate temporal popularity, into the content feature set. We continuously update these dynamic features in the delivery phase by aggregating users' interactions over content items in a real-time manner. We demonstrate that maintaining content profiles with dynamic features is an effective strategy to overcome the cold-start problem on dynamic content.

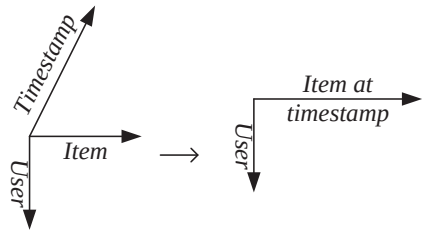

Figure 1: An illustration of unfolding a multi-dimensional event.

The open profiles of users provide valuable information about user preferences and interests that helps in recommending content for new users. Historical feedback given by users on content of interest, such as ratings or click stream, directly reveals users' opinion on the content universe. The bilinear regression models we proposed can discover association patterns between the general user profiles and the content features by exploiting the interactive data (the typical user profile in traditional CF). The established associations are then applied to evaluating individualized appraisement over currently available items for accurate and prompt personalized recommendations in real time.

This work is motivated by a personalized content optimization task for the Today-Module on Yahoo! Front Page. The effectiveness of the bilinear models is verified on a large-scale real-world data set collected in the application. This approach results in an offline model except online tracked dynamic features in content profiles. The computational overhead in online recommendation is minor compared with recommender systems that require online re-training. The framework is general and flexible, which can be adapted to other personalized tasks.

The paper is organized as follows: We introduce data representation in Section 2, which includes content profiling, user profiling and interactive feedback; In Section 3 we describe a family of probabilistic bilinear models in detail that covers training algorithms and further discussions on potential capabilities; We review related work in Section 4; We report the experimental results on the data set collected from the Today-Module with comparison against six competitive alternatives in Section 5 and conclude in Section 6.

2. DATA REPRESENTATION

The observational data is naturally recorded in multi-dimensional format. A logistic event is associated with at least three types of objects, $user \times content \times timestamp$. These multi-dimensional events can always be flattened into two-way form without loss of generality, see Figure 1 for an illustration. In personalization on dynamic content, we can treat $content \times timestamp$ as items of interest. Note that the dimension of *timestamp* is usually not considered in traditional recommender systems. The flattened dimensions form a new content item space, in which features are extracted for profiling. We generate and maintain three sets of data: content profiles, user profiles, and interactive feedback on content items of interest.

2.1 Content Profiles

When a content is either created or acquired, the informa-

tion related to the content, such as manufacturer, product name and categories etc., constitutes an initial part of the profile. Continuous refinement of the content profile helps to optimize the use of the content assets. In the delivery phase, the content is delivered to users and interactions on the content are logged and analyzed, providing the ability to assess the content popularity in a real-time manner. The content popularity over time is a crucial ingredient in content management, since the commercial value of most content is varying or decaying temporally, especially for breaking news.

We consider generalized content items here, which are redefined with both temporal characteristics and other conditions. In a content profile, there are at least two groups of features:

- Static descriptors: Such as categories, manufacturer name, title, bag of words of textual content etc.
- Temporal characteristics: Such as popularity, click-through rate (CTR) and price at current time stamp or the hours elapsed after content acquisition.

We can collect any features related to the content items. For example, in search the items become webpages fused with a query, and then joint features, such as contextual co-occurrences, can be constructed.

Each content is represented as a vertical vector, denoted by $\mathbf{z}$, where $\mathbf{z} \in \mathbb{R}^C$ and C is the number of content features.

2.2 User Profiles

The objective of collecting visitor information is to develop a user profile that describes a site visitor's interests, consumption history, and other descriptors important to the site owner. A review of various user profiling techniques is provided in [19]. Explicit profiling requests each visitor to declare personal information, such as age, gender and occupation, or to fill out questionnaires that explicitly state their preferences. Implicit profiling tracks the visitors' behavior and it is generally transparent to the visitor. Browsing and purchasing patterns are the behaviors most often assessed. The profile combined with demographic, transaction, and navigation data implicitly represents a user's preferences and recent interests.

The user feature space is spanned by legally usable features. Each user is represented as a vertical vector, denoted by $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^D$ and D is the dimensionality of the user feature space.

2.3 Interactive Feedback

In traditional collaborative filtering (CF), the feedback given by users on content of interest are used as user profiles to evaluate commonalities between users. In our regression approach, we separate the feedback from user profiles. The feedback on content of interest is utilized as targets that relate patterns in user features to content features.

Although the interactions between the users and the available items vary depending on the types of items involved, we can always observe or measure some feedback from user side. For example, a user may purchase a product or a service after review, and even rate it later. For a content posted on a Web page, a user may click to see more details. The ratings and actions (click or not, purchase or not) provide explicit feedback.[1] There are a range of efforts attempted to measure various kinds of implicit feedback indicators from linger time [13] to eye movements [36]. We focus on two types of feedback in this paper:

- Continuous scores: most implicit feedback and ratings can be converted as continuous scores.
- Binary actions: such as click or not, purchase or not after reviewing an item.

We have collected three sets of data, including content features, user profiles and interactive data between users and items. Let index the i-th user as $\mathbf{x}_i$ and the j-th content item as $\mathbf{z}_j$, and denote by r_{ij} the interaction between the user $\mathbf{x}_i$ and the item $\mathbf{z}_j$. We only observe interactions on a small subset of all possible user/item pairs, and denote by $\mathbb{O}$ the set of observations $\{r_{ij}\}$.

3. BILINEAR REGRESSION MODELS

The user and content profiles provide timely descriptions of users and items respectively. As the two feature spaces are usually dichotomous, it is hard to apply the contextual data mining techniques [9] here. However, the interactive feedback reveals the correlations between user patterns and content features. In this section, we describe a family of predictive bilinear models to discover pattern affinities between heterogeneous features. A set of weight coefficients is introduced to capture the pairwise associations between user and content features. The parametric model is optimized by fitting the observed interactive feedback.

3.1 Bilinear Indicator

The bilinear models can be regarded as a special case in the Tucker family [14], which have been widely applied in machine learning applications. For example, Tenenbaum and Freeman [39] developed a bilinear model for separating "style" and "content" in images, and recently Chu and Ghahramani [11] derived a probabilistic framework of the Tucker family for modeling structural dependency from partially observed high-dimensional array data.

We define an indicator as a bilinear function of $\mathbf{x}_i$ and $\mathbf{z}_j$ in the following:

$$s_{ij} = \sum_{a=1}^{C} \sum_{b=1}^{D} x_{i,b} z_{j,a} w_{ab}, \tag{1}$$

where D and C are the dimensionality of user and content features respectively, $z_{j,a}$ denotes the a-th feature of $\mathbf{z}_j$ and $x_{i,b}$ denotes the b-th feature of $\mathbf{x}_i$. The weight variable w_{ab} is independent of user and content features and quantifies the affinity of these two factors $x_{i,b}$ and $z_{j,a}$ in interactions.[2]

The scalar s_{ij} is generated by mixing these basis vectors with coefficients given by the Kronecker product of $\mathbf{x}_i$ and $\mathbf{z}_j$. The indicator can be equivalently rewritten as

$$s_{ij} = \mathbf{w}^\top (\mathbf{z}_j \otimes \mathbf{x}_i),$$

[1] Clicks and user purchase history are often considered as implicit feedback in other collaborative filtering literature since these may not reflect real user preferences. For example, a user may find that an article is uninteresting after clicking and reading it. However, we refer these actions as explicit feedback since the user intentions of these actions are clearer than those of other implicit feedback such as linger time and eye movement.

[2] In practice, we also insert an individual-specific offset for

where $\mathbf{w}$ is a column vector of entries $\{w_{ab}\}$, and $\mathbf{z}_j \otimes \mathbf{x}_i$ denotes the Kronecker product of $\mathbf{x}_i$ and $\mathbf{z}_j$, a column vector of entries $\{x_{i,b}z_{j,a}\}$. In matrix form, eq(1) can be rewritten as

$$s_{ij} = \mathbf{x}_i^\top \mathbf{W}\, \mathbf{z}_j,$$

where $\mathbf{W}$ denotes a $D \times C$ matrix with entries $\{w_{ab}\}$, which describes a linear projection from the user feature space onto the item feature space. The projected user profile $\mathbf{W}^\top \mathbf{x}_i$ is aligned to the item features, denoted by $\tilde{\mathbf{x}}_i$, which can be explained as users' preferences on item characteristics accordingly. Then the indicator becomes a dot product, i.e. $s_{ij} = \tilde{\mathbf{x}}_i^\top \mathbf{z}_j = \sum_{a=1}^{C} \tilde{\mathbf{x}}_{i,a} z_{j,a}$.

To further examine the feature functions, let us distinguish dynamic features in the item feature vector as $\mathbf{z}_j = \begin{bmatrix} \mathbf{z}_{j,s} \\ \mathbf{z}_{j,d} \end{bmatrix}$, where $\mathbf{z}_{j,s}$ denotes static features and $\mathbf{z}_{j,d}$ denotes dynamic features that vary along time. The indicator s_{ij} can then be rewritten as follows,

$$s_{ij} = \left[\mathbf{x}_i^\top \mathbf{W}_s, \mathbf{x}_i^\top \mathbf{W}_d\right] \begin{bmatrix} \mathbf{z}_{j,s} \\ \mathbf{z}_{j,d} \end{bmatrix} = \tilde{\mathbf{x}}_{i,s}^\top \mathbf{z}_{j,s} + \tilde{\mathbf{x}}_{i,d}^\top \mathbf{z}_{j,d}, \quad (2)$$

where $\mathbf{W}_s$ and $\mathbf{W}_d$ denote the columns in $\mathbf{W}$ associated with the static and dynamic item features respectively, and $\tilde{\mathbf{x}}_{i,s}$ and $\tilde{\mathbf{x}}_{i,d}$ denote the i-th user's preferences on the static and dynamic item features respectively.

Note that a user's score s_{ij} on an item is composed of three parts: $\tilde{\mathbf{x}}_{i,s}^\top \mathbf{z}_{j,s}$ reflects long-term personal preferences on content features learnt from historical activities; $\mathbf{z}_{j,d}$ is of dynamic characteristics, in our work which include temporal popularity over the whole user population, i.e. article quality; the tradeoff between static personal preferences and article quality is determined by $\tilde{\mathbf{x}}_{i,d}$.

On cold-start with new items, the user's preferences on static item features $\tilde{\mathbf{x}}_{i,s}^\top \mathbf{z}_{j,s}$ play an important role, as the dynamic features couldn't be accurately estimated at the beginning stage. Similarly, on cold-start with new users, recommendations are fully determined by the users' preferences on content features $\tilde{\mathbf{x}}_i$, which are projected from the user profile $\mathbf{x}_i$.[3] As we will show in the following, the projection $\mathbf{W}$ can be learnt from the historical interactive feedback.

3.2 Probabilistic Framework

We employ appropriate likelihood functions to relate the indicator s_{ij} to different types of observed interactions.

- Continuous scores with Gaussian measurement noise:

$$p(r_{ij}|s_{ij}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r_{ij} - s_{ij})^2}{2\sigma^2}\right),$$

  where σ stands for the noise level.[4]

- Binary actions with $r_{ij} \in \{-1, 1\}$. The logistic function is widely used as the likelihood function, which is defined as

$$p(r_{ij}|s_{ij}) = \frac{1}{1 + \exp(-r_{ij} s_{ij} + \gamma)},$$

  where γ denotes a bias term, usually set at 1.

Given a set of $\mathbf{w}$, the likelihood of observing the interactive data can be evaluated by

$$p(\mathbb{O}|\mathbf{w}) = \prod_{ij} p(r_{ij}|s_{ij}), \quad (3)$$

where the index ij runs over the observational set $\mathbb{O}$.

We also specify a standard Gaussian distribution over the weight variables as a priori,

$$p(\mathbf{w}) = \frac{1}{\sqrt{2\pi}\varsigma} \exp\left(-\frac{\sum_{ab} w_{ab}^2}{2\varsigma^2}\right), \quad (4)$$

where ς^2 is the variance.

Based on the Bayes' theorem, the posterior distribution of $\mathbf{w}$ is proportional to the product of the likelihood and the prior,

$$p(\mathbf{w}|\mathbb{O}) \propto p(\mathbb{O}|\mathbf{w})\, p(\mathbf{w}). \quad (5)$$

where $p(\mathbf{w})$ is the prior distribution defined as in eq(4) and $p(\mathbb{O}|\mathbf{w})$ is the likelihood defined as in eq(3).

3.3 Offline Modeling

In this section, we describe a training algorithm in batch mode to estimate the posterior distribution of the weight coefficients $p(\mathbf{w}|\mathbb{O})$ as in eq(5). For continuous scores with Gaussian noise, the posterior distribution is still a Gaussian due to the conjugate property. With non-Gaussian likelihood functions, the posterior distribution becomes non-Gaussian. However we can always approximate the true distribution by a Gaussian distribution. One of the most popular techniques is the Laplace approximation [26], which finds the mode of the true posterior as the approximate mean and approximates the inverse covariance matrix by the Hessian matrix, the second order derivatives with respect to the weights at the mode point.

The mode, also known as the maximum-a-posteriori (MAP) estimate, can be found by maximizing the joint probability $p(\mathbb{O}|\mathbf{w})p(\mathbf{w})$. The optimization problem is equivalent to minimizing the negative logarithm of the joint probability, i.e.

$$\min_{\mathbf{w}} L(\mathbf{w}) = \frac{1}{2\varsigma^2} \sum_{ab} w_{ab}^2 - \sum_{ij} \log p(r_{ij}|s_{ij}), \quad (6)$$

where ς^2 plays a role of tradeoff. The gradient with respect to w_{ab} can be computed as follows,

$$\frac{\partial L(\mathbf{w})}{\partial w_{ab}} = \frac{w_{ab}}{\varsigma^2} - \sum_{ij} \frac{\partial \log p(r_{ij}|s_{ij})}{\partial s_{ij}} x_{i,b} z_{j,a}, \quad (7)$$

and gradient-decent packages can then be employed to find the minimum. Note that the objective functional is convex and the minimum is unique. The detailed formulations are given in Table 1 and the gradient-descent algorithm is summarized as in Table 2. Each objective/gradient evaluation costs $\mathcal{O}(NCD)$, where CD is the size of $\mathbf{w}$ and N is the size of the observed set $\mathbb{O}$. Note that matrix inverse can

each user. The final scalar is evaluated as

$$s_{ij} = \sum_{a=1}^{C} \sum_{b=1}^{D} x_{i,b} z_{j,a} w_{ab} + \mu_i,$$

where $\mu_i \in \mathbb{R}$ denotes a user-specific offset. Here μ_i is used to tradeoff the user's activity level, since some users are active clickers while some are casual users.

[3]There is an implicit assumption that the user profile is rich enough to be transformed into preferences on item characteristics. This condition can be easily satisfied in practice.

[4]In practice, the noise level could be prefixed at an appropriate value based on the signal/noise ratio.

Table 1: The logarithm likelihood functions and the first-order derivatives.

Target	$\log p(r_{ij}\|s_{ij})$	$\frac{\partial \log p(r_{ij}\|s_{ij})}{\partial s_{ij}}$
Continuous	$-\frac{(r_{ij}-s_{ij})^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma)$	$\frac{s_{ij}-r_{ij}}{\sigma^2}$
Binary	$-\log(1+\exp(-r_{ij}s_{ij}+\gamma))$	$r_{ij}p(-r_{ij}\|s_{ij})$

Table 2: The gradient-descent algorithm for MAP.

1.	Initialize $\mathbf{w}=0$, given σ^2 and ς^2
2.	While objective/gradient evaluation at $\mathbf{w}$ is requested: Compute the objective as in eq(6); Compute the gradients for $\mathbf{w}$ as in eq(7); Return the objective/gradients to the package.
3.	Until the optimization package returns the final $\mathbf{w}$.

be applied directly to the case of continuous targets for an solution, but the computational cost is $\mathcal{O}(NC^2D^2+C^3D^3)$. It is very expensive for the cases having a large number of features.

3.4 Prediction

The MAP estimate, denoted as $\mathbf{w}^{\text{MAP}}$, is then applied to new user/item pairs for prediction. For any pair of $\mathbf{x}_i$ and $\mathbf{z}_j$ in test, the best guess of the indicator s_{ij} is determined as follow,

$$\hat{s}_{ij} = \sum_{a=1}^{C}\sum_{b=1}^{D} x_{i,b} z_{j,a} w_{ab}^{\text{MAP}}, \tag{8}$$

where w_{ab}^{MAP} is an entry of the MAP estimate $\mathbf{w}^{\text{MAP}}$.

3.5 Discussions

In this section, we discuss model selection and some potentials of the framework we proposed, such as online learning and active learning.

3.5.1 Model Selection

The prior variance ς^2 is an important model parameter in the regression framework. The most common approach in practice to determine the best model setting is cross validation. In k-fold cross validation, the original training data is randomly partitioned into several folds, whereas in our application having time series of dynamical features we have to split the training data by a temporal point into two folds, usually with size ratio $2:1$. Given a particular set of model parameters, we run the training algorithm on the fold of earlier data to estimate the weight coefficients, and test the resulting model on the left-out fold to obtain the validation error. The predictive performance indicates the goodness of the model parameter setting. We try grid search over a set of parameter values to find the optimal one on which we observe the best performance on the validation data. The optimal weight coefficients in the regression model are finally obtained by training on the whole training data set using the best set of model parameters.

3.5.2 Online Learning and Active Learning

In this work we only focus on training an offline model coupled with dynamic features, whereas the probabilistic framework we employed provides the capacity of online learning as well. Assumed-density filtering (ADF) is a one-pass, sequential method for computing an approximate posterior distribution [17]. In ADF, observations are processed one by one, updating the posterior distribution which is usually approximated as a Gaussian before processing the next observation. The approximate posterior is found by minimizing KL-divergence to preserve a specific set of posterior expectations. Recently, Expectation Propagation [29] extends ADF to incorporate iterative refinement of the approximations, which iterates additional passes over the observations and does not require corresponding with time of arrival as in time series.

Learning could be made more efficient if we can actively select salient data points. Within the probabilistic regression framework, the expected informativeness of a new observation can be measured by the change in entropy of the posterior distribution of the weight coefficients after inclusion of the candidate [24]. The new posterior distribution with the inclusion of the unused sample can be approximated as a Gaussian by ADF-like online learning algorithms. Based on information-theoretical principles, the entropy gain on the posterior distribution of weight variables can then be applied as the criterion for candidate election.

4. RELATED WORK

Our work is closely related to adaptive news systems, one of the most popular types of personalized Web-based service [6]. The most relevant previous work to our study would be the Google News recommender system [15], a content-agnostic system which combines three different algorithms using a linear model to generate recommendations in News domain. However, since the proposed approach is a pure collaborative filtering, it does not solve the cold-start problem for new users. Even though ratings from new users can be updated in near real-time by gridifying their algorithm, it still needs to wait until new users provide ratings or clicks before making recommendations. Also, the reported results are based on two heavy user data sets (top 5K heavy users with 370K clicks and 500K users with 10M clicks), where effects of new and casual users haven't been considered. In our application of the Today-Module on Yahoo! Front Page, 40% of clickers are new clickers with no historical clicks, 82% of clickers have less or equal to 5 historical clicks, 92% of clickers have no more than 10 historical clicks as shown in the Figure 3(b). Another key difference lies in that Google News [15] is a content-agnostic system which doesn't resort to either content features or user information. YourNews [2] allows users to customize their interest profiles through a user model interface. The study on user behavior shows the benefit from customization but also cautions the downside on system performance. In our application, we build up user and content profiles without any solicitation on users. Newsjunkie [18] provided personalized news feeds for users by measuring news novelty in the context of stories the users have already read. Our content profiles can also maintain dynamic features in addition to context novelty, such as popularity and freshness. Our model also leverages user profiles to facilitate cold-start on new users.

Our work is also related to personalized search, though the tasks are quite different. Micarelli et al. [28] gave a nice review on this direction. Personalized search builds models of short-term and long-term user needs based on observed user actions, which is able to satisfy the users better than standard search engines based on traditional Information Retrieval (IR) techniques. Speretta and Gauch [38] devel-

oped user profiles from their query histories and used these profiles to re-rank the results returned by an independent search engine by giving more importance to the documents related to topics contained in the user profile. Ahn et al. [2] designed the TaskSieve system that utilizes a relevance feedback-based profile for personalized search. Both systems employ the traditional linear approach to combine personal preferences and query relevance. The combined score is calculated as $\alpha f(\mathbf{x}_i, \mathbf{z}_j) + (1-\alpha)\, r(\mathbf{z}_j)$, where $\mathbf{x}_i$ is a user profile, $\mathbf{z}_j$ is a content item fused with the query and α is the tradeoff. There is a strong correspondence to the terms in eq(2). By replacing the dynamic features of $\mathbf{z}_j$ by the query relevance $r(\mathbf{z}_j)$ and implement $f(\mathbf{x}_i, \mathbf{z}_j)$ in the parametric form of the long-term preferences as in eq(2), our bilinear model provides a flexible framework to learn the personal preference function and the tradeoff term from the click stream in a principled manner.

A personalized service may not be exactly based on individual user behaviors. The content of a website can be tailored for a predefined audience, based on offline research of conjoint analysis, without online gathering knowledge on individuals for service. Conjoint analysis is one of the most popular market research methodologies for assessing how customers with heterogeneous preferences appraise various objective characteristics in products or services. Analysis of tradeoffs driven by heterogeneous preferences on benefits derived from product attributes provides critical inputs for many marketing decisions, e.g. optimal design of new products, target market selection, and pricing a product. In very early studies [40], homogeneous groups of consumers are entailed by the use of a priori segmentation. For example, consumers are assigned to groups on the basis of demographic and socioeconomic variables, and the conjoint models are estimated within each of those groups. This is closely related to demographic recommender systems [23, 34], in which recommendations are based on demographic classes categorized by users' personal attributes. However, the criteria in the two steps are not necessarily related: one is the homogeneity of customers in terms of their descriptor variables and another is the conjoint preferences within segments. Traditionally, conjoint analysis procedures are of two-stage: 1) estimating a parametric function which represents customers' preference at individual-level in terms of user profiles, e.g. hierarchical Bayesian methods [25]; 2) through clustering algorithms, grouping users into segments where users share similar individual-level preferences. Jiang and Tuzhilin [22] experimentally demonstrated both 1-to-1 personalization and segmentation approaches significantly outperform aggregate modeling.

In the extreme cold-start setting with dynamic content and a large amount of new users, traditional collaborative filtering methods cannot provide recommendation effectively. A number of hybrid methods, which combine information filtering and other collaborative filtering techniques, have been proposed, such as [12] of an online newspaper and the Fab system [3]. Good et al. [20] improved accuracy by introducing personal agents, and Park et al. [33] further improved its performance in cold-start situations by adding small number of artificial users who have rated all items. However, this approach performs better only if a user has rated a few items but does not solve the cold-start problem directly for new users. Basu et al. [5] utilized social information in content-based filtering. Melville et al. [27] em-

Figure 2: A snapshot of the default "Featured" tab in the Today Module on Yahoo! Front Page. There are four articles displayed at footer positions. One of the four articles is highlighted at the story position.

ployed a content-based predictor to enhance existing user data, and then provided personalized suggestions through collaborative filtering. Basilico and Hofmann [4] developed a framework that incorporates all available information by using a suitable kernel or similarity function between user-item pairs. Hybrid methods are especially useful when data is sparse, for example in cold-start situations [35], but to our best knowledge none of previous work has been integrated with continuous online attributes, such as popularity or freshness.

5. CASE STUDIES

In this section, we verify the capacity of the proposed bilinear models on a real-world application. We start with an introduction of the problem settings in Yahoo! Today-Module and describe the attributes we collected in user/content profiling. We also define performance metrics to evaluate predictive results and report experimental results with comparison to competitive approaches.

5.1 Yahoo! Today Module

Today-Module is the most prominent panel on Yahoo! Front Page, which is also one of the most visited pages on the Internet, see a snapshot in Figure 2. The default "Featured" tab in Today Module highlights one of four high-quality articles, mainly news, while the four articles are selected from a daily-refreshed article pool curated by human editors. As illustrated in Figure 2, there are four articles at footer positions, indexed by F1, F2, F3 and F4 respectively. Each article is represented by a small picture and a title. One of the four articles is highlighted at the story position, which is featured by a large picture, a title and a short summary along with related links. *At default, the article at F1 is highlighted at the story position.* A user can click on the highlighted article at the story position to read more details if she is interested in the article. The event is recorded as a story click. If a user is interested in an article at F2∼F4 positions, she can highlight the article at the story position by clicking on the footer position. To draw visitors' attention, we would like to rank available articles according to visitors' interests, and highlight the most attractive article at F1 position.

It is difficult to adopt a traditional collaborative filtering algorithm such as user-user [7] or item-based [16] in the

Today-Module. Retrieving historical ratings of users for similarity evaluation in the online service is hard. Lifetime of an article is very short (only a few hours) and old articles will be pulled out of content pool regularly. Another difficulty is that we always need to recommend new items and significant portion of users is taken by new users. As shown in Figure 3(b) and (c), 40% clickers in the test data are the first time clickers without any historical clicks, and on average 60% articles are new everyday. Thus, traditional collaborative filtering methods suffer from the cold-start problem. Furthermore, the article popularity is temporally decaying, see Figure 3(a) for an example where CTR decreases to 1/6 of its peak value at the end of the article's lifetime. It is difficult to compare users' feedback on the same article received at different time slots.

5.2 Data Collection

We collected events from a random bucket in July 2008. In the random bucket, articles are randomly selected from the content pool to serve users. An event records a user's action on the article at the story position, which is either "view" or "click" encoded as -1 and 1 respectively. Note that a user may click on the same article multiple times but at different time slots.[5] In our approach, these binary events are distinguishable, because a content item is defined by both the article and the time slot of the event of interest, see Figure 1. This is a conceptual difference from traditional approaches.

We collected about 40 million click/view events by about 5 million users from the random bucket before a certain time stamp for training. We also collected about 0.6 million click events after that time stamp for test.

The features of users and items were selected by "support". The "support" of a feature means the number of users having the feature. We only selected the features of high support above a prefixed threshold, e.g. 10% of the population. Then each user is represented by a vector of more than one thousand categorical features, which include:

- Demographic information: gender (2 classes) and age discretized into ten classes;
- Geographic features: about two hundred locations of countries or U.S. States;
- Behavioral categories: about one thousand binary categories that summarize the user's consumption behavior within Yahoo! properties;

Each article is profiled by a vector of about one hundred static features and a *dynamic feature*. The static features include:

- URL categories: tens of classes inferred from the URL of the article resource;
- Editor categories: tens of topics tagged by human editors to summarize the article content;

The *dynamic feature* is of estimated click-through rate (CTR) at events of interest, which differentiates the same article at different time slots. We adapted the Kalman filter designed and implemented by our team [1] for CTR tracking, which yields a good indicator of article quality and popularity temporally. Note that other dynamic features, e.g. freshness, can be added into the content profiles as well.

[5]For anti-robot purpose, the number of clicks generated by one user on the same article could be at most 1 within a single time slot (e.g. 5 minutes).

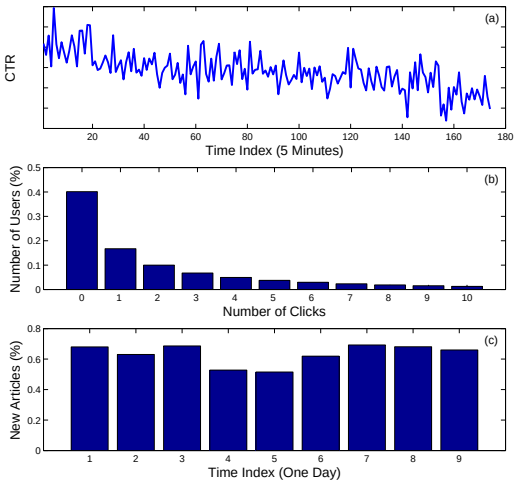

Figure 3: (a) A typical pattern of article CTR; (b) Historical click counts of clickers in test; (c) New article percentage per day in test.

Categorical features are encoded as binary vectors with non-zero indicators. For example, "gender" of two classes is translated into two binary features, i.e., "male" is encoded as $[0, 1]$, "female" is encoded as $[1, 0]$ and "unknown" is $[0, 0]$.[6] As the number of non-zero entries in these binary feature vectors varies, we further normalized each vector into unit length, i.e., non-zero entries in the normalized vector are replaced by $1/\sqrt{k}$, where k is the number of non-zero entries. For user features, we normalized behavioral categories and the remaining features (age, gender and location) separately, due to the variable length of behavioral categories per user. For article features, we normalized URL and Editor categories together, and kept the CTR term (a real value) intact. Following conventional treatment, we also augmented each feature vector by a constant term 1. Each content item is represented by a feature vector of 83 entries, while each user is represented by a feature vector of 1193 entries.

5.3 Performance Metric

For each user in test, we computed predictive scores as in eq(8) for all available articles at the time stamp of the event, and ranked these articles in descending order according to the scores. On click events, we measured the rank position of the article being clicked by the user.

The first metric we use is the number of clicks in each rank position. A good predictive model should have more clicks on the top-ranked positions and lesser clicks on the lower-ranked position. In our application, we mainly concern the performance on the top 4 positions.

We also proposed a simple utility function to quantify the predictive performance, which is defined as follows:

$$U = \sum_{r=1}^{4} \frac{U_r}{2^{r-1}}, \tag{9}$$

[6]The "unknown" category coded with zero entries has little contribution to our linear models.