LogUCB: An Explore-Exploit Algorithm For Comments Recommendation*

Dhruv Mahajan
Microsoft Research India
Bangalore
dhrumaha@microsoft.com

Rajeev Rastogi
Amazon
Bangalore, India
rastogi@amazon.com

Charu Tiwari
Yahoo! Labs
Bangalore, India
charu@yahoo-inc.com

Adway Mitra
CSA Department, IISc
Bangalore, India
adway@csa.iisc.ernet.in

ABSTRACT

The highly dynamic nature of online commenting environments makes accurate ratings prediction for new comments challenging. In such a setting, in addition to *exploiting* comments with high predicted ratings, it is also critical to *explore* comments with high uncertainty in the predictions. In this paper, we propose a novel *upper confidence bound* (UCB) algorithm called LOGUCB that balances exploration with exploitation when the average rating of a comment is modeled using logistic regression on its features. At the core of our LOGUCB algorithm lies a novel variance approximation technique for the Bayesian logistic regression model that is used to compute the UCB value for each comment. In experiments with a real-life comments dataset from Yahoo! News, we show that LOGUCB with bag-of-words and topic features outperforms state-of-the-art explore-exploit algorithms.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Learning

General Terms

Algorithms, Experimentation

Keywords

logistic regression, explore-exploit, comment ratings, upper confidence bound

1. INTRODUCTION

User generated content in the form of comments has witnessed an explosive growth on the web. Web sites like Yahoo! and YouTube allow users to comment on diverse content like news articles and videos. These "crowd-sourced" comments are highly engaging because they reflect the views and opinions of real users, and are a means for users to discuss and debate controversial issues. Furthermore, comments can be informative, and are an effective gauge of public sentiment on a topic.

However, a key challenge with user generated comments is that a single news article on a major event can easily trigger thousands of comments. Clearly, it is unrealistic to expect that users will sift through the copious comments for an article. A more realistic scenario is that a user will look through the first K comments that are presented to him/her, and ignore the remaining comments. Fortunately, most commenting environments allow users to provide feedback on comments shown to them – users can give a *thumbs-up* rating to indicate that they like, or a *thumbs-down* rating to express dislike for a comment. Thus, our goal in this paper is to develop algorithms that leverage past comment ratings to rank an article's comments so that the top K comments have the highest chance of being liked by the user.

Recommender systems have been extensively studied in the research literature – these systems use collaborative filtering, content-based filtering, or hybrid approaches to recommend items to users. Traditional collaborative filtering techniques rely on either item-item/user-user similarity models [26], or matrix factorization models [22] that multiply user- and item-specific factors for ratings prediction. A shortcoming of classical collaborative filtering approaches is that they cannot predict ratings for new users or items (referred to as the *cold-start problem*). Content-based filtering methods alleviate the cold-start problem by building predictive models based on user and item features like age, gender, and category. But they do not consider user- or item-specific parameters learnt from past ratings. Recently proposed hybrid approaches [2, 1, 4] combine collaborative filtering with content-based filtering by simultaneously incorporating features and user/item-centric parameters in models – this allows them to predict ratings more accurately for both existing user-item pairs as well as new pairs (through features). The top K items with the highest predicted ratings are then recommended to the user.

Commenting environments are highly dynamic with new comments and articles arriving continuously. These new comments have zero or few user ratings, and constantly changing content that may have little overlap with previous comments. Furthermore, of the users who view comments, only a small fraction typically rate comments causing training data to be extremely sparse. In such a scenario, even models that exploit features and past ratings in-

*Work done while the authors were at Yahoo!.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

formation may be unable to make accurate predictions, and there may be a high degree of uncertainty in the predicted ratings of new comments. Consequently, simply ranking comments based on their predicted rating scores may lead to poor comments being ranked high and good comments being ranked low because of inaccuracies in their rating estimates. Thus, a strategy that only *exploits* user comments with high predicted ratings may be suboptimal, and it is important to also *explore* comments with low predicted ratings but high uncertainty since their true ratings may be much higher.

Our comments recommendation problem is an instance of the well-known *exploitation-exploration* tradeoff that has been studied extensively in the context of the *multi-armed bandits* problem [12, 6]. Given a set of arms with unknown rewards, the objective in bandit problems is to select arms in successive trials (one per trial) so that the expected total reward over a finite number of trials is maximized. In our comments setting, comments correspond to arms and the ratings they receive are the rewards. Existing bandit algorithms [6, 16, 7] balance exploration and exploitation to achieve optimal rewards. Exploration gathers information to refine the expected reward estimates for seemingly suboptimal arms while exploitation selects arms with high reward estimates based on past observations.

A popular class of bandit algorithms are the *upper confidence bound* (UCB) algorithms [6, 16, 7] – these compute an upper confidence bound for each arm which is essentially a high-probability upper bound on the expected reward. The UCB for an arm is computed by combining the current reward estimate with the uncertainty in the estimate. In each trial, the arm with the highest UCB value is then selected. Thus, the selected arms either have a high reward estimate (and so should be exploited more to increase the total reward) or high uncertainty (and so should be explored more since the arm could potentially give a high reward). In recent work, [16] proposes a UCB-based bandit algorithm LINUCB that estimates the reward of each arm through a linear regression on features. Leveraging textual and other informative features help to overcome data sparsity and improve the accuracy of reward estimates. The authors compute the variance in the reward estimated by the linear model and use it to determine the UCB for each arm. Such bandit algorithms that utilize context information (e.g., features) are called *contextual bandits*.

In this paper, we propose a UCB-based contextual bandit algorithm LOGUCB that estimates average ratings for comments using *logistic regression* on features. Logistic regression is better suited for modeling thumbs-up/down style binary ratings data compared to linear regression. Unlike linear regression, it ensures that average ratings lie in the fixed range [0, 1], and so estimates have bounded variance (between 0 and 1). Moreover, logistic regression models are more robust to outliers [8]. We adopt a Bayesian approach to estimate average ratings for comments and the variance in the estimates. Since logistic regression models are more complex, the mean rating estimates and variance are not available in closed form, and need to be approximated. Similar to UCB-type algorithms, we combine the rating estimate (exploitation) with the variance in the estimate (exploration) to compute a UCB for each comment, and select the top-K comments with the highest UCBs.

Our main contributions can be summarized as follows:

1) We propose a novel UCB-based algorithm called LOGUCB that balances exploration with exploitation when the average rating of a comment is a logistic sigmoid function of its features. Our LOGUCB algorithm for recommending comments consists of the following core modules: (1) A global logistic regression model that uses past comments to compute a global prior on feature weights, (2) A per-article Bayesian logistic regression model that incorporates the global prior and additional slack variables per comment – in the beginning when ratings data is sparse, the model relies on past comments and comment features for rating prediction; but once a comment has sufficient ratings, the model makes a smooth transition and uses empirical rating estimates, and (3) An explore-exploit module that combines the predicted rating with uncertainty in the prediction to determine the comments recommended to the user.

2) To compute the UCB for each comment, we assume a Bayesian setting and employ a host of approximation techniques to efficiently estimate the mean rating and associated variance. Our variance approximation technique for the Bayesian logistic regression model is novel and lies at the core of our LOGUCB algorithm – this is a key technical contribution of our work.

3) In addition to the words in each comment, we also include a comment's *Latent Dirichlet Allocation* (LDA) topics [10] as features to further improve rating prediction accuracy.

4) We conduct an extensive experimental study on a real-life dataset containing comments for Yahoo! News articles. Our experimental results indicate that our per-article logistic regression models incorporating global priors and leveraging bag-of-words and topic features predict ratings with higher accuracy compared to simple baselines. Furthermore, our LOGUCB explore-exploit algorithm outperforms state-of-the-art techniques like LINUCB.

2. SYSTEM MODEL AND PROBLEM DEFINITION

Each online article has an associated set of user generated comments. Every time an article is viewed by a user, our system selects K comments for the article to display to the user. The user then assigns one of two ratings to a (random) subset of the K comments shown to him/her: (1) A *thumbs-up* rating if the user likes the comment, or (2) A *thumbs-down* rating to express dislike for the comment. The user can also post additional comments for the article. Thus, the comments for an article and the ratings for each comment are highly dynamic.

Consider an article a at time t. We denote the set of comments posted by users for article a at time t by $C(t)$ and the number of comments in $C(t)$ at time t by $N(t)$. Thus, $N(t) = |C(t)|$. Each comment $i \in C(t)$ has an associated M-dimensional feature vector $\mathbf{x}_i$. The M feature values for a comment are computed based on its content and include word occurrence frequency, comment length, presence of numeric values or special symbols like exclamation marks, etc.

We will use binary 0/1 values for ratings with a 1 corresponding to a thumbs-up, and a 0 to a thumbs-down. We will denote by $n_i(t)$ the total number of 0/1 ratings provided by users for comment i at time t – of these, the number of 0 and 1 ratings are denoted by $n_i^-(t)$ and $n_i^+(t)$, respectively. Furthermore, we will represent the vector of observed ratings for comment i by $\mathbf{y}_i(t)$ and the j^{th} rating for comment i by y_{ij}. Finally, we will use $\mathbf{X}(t) = \{\mathbf{x}_i\}$ and $\mathbf{Y}(t) = \{\mathbf{y}_i(t)\}$ to denote the feature and rating vectors, respectively, for all the comments of article a in $C(t)$.

Let μ_i be the expected rating of comment i, that is, $\mu_i = E[y_{ij}]$. Note that μ_i for comment i is unknown. At time t, our comments recommendation system only has access to the comments $C(t)$ and rating information $\mathbf{Y}(t)$ for the comments – based on these, it tries to identify the K best comments with the highest expected ratings, and recommends them to users. These comments with high likeli-

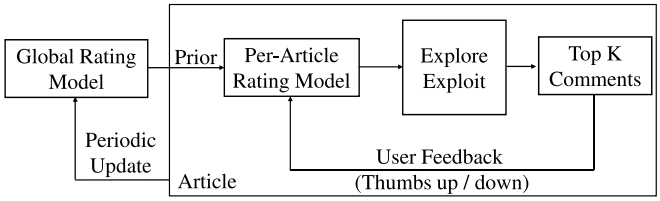


Figure 1: LOGUCB **Components.**

hoods of getting a thumbs-up are precisely the ones that the average user will like, and thus will lead to increased user satisfaction.

Comments Recommendation Problem: Let article a be viewed at times $t_1, t_2, t_3 \ldots$. At each time t_l, the problem is to select K comments from $C(t_l)$ to recommend such that the total expected number of thumbs-up ratings received until time t_l is maximum, that is, $\sum_{i \in C(t_l)} \sum_{j=1}^{n_i(t_l)} E[y_{ij}]$ is maximum. □

As mentioned earlier, our comments recommendation problem above has a natural mapping to the *multi-armed bandit* problem with comments as arms, and ratings as the corresponding rewards.

3. LOGUCB ALGORITHM COMPONENTS

LOGUCB is a novel UCB-style contextual bandit algorithm that employs logistic regression on comment features to model the expected rating for a comment. For each article a, LOGUCB trains a separate logistic regression model. We consider a Bayesian setting in our work because the priors on model parameters help to regularize them while the posteriors provide a natural method to compute the variance of estimated ratings.

Below, we give a brief description of the different modules of our LOGUCB algorithm shown in Figure 1.

Global Rating Model: A newly posted article has very few comment ratings initially. Consequently, the per-article model that we train can lead to overfitting due to data sparseness issues. To address this, we learn a global logistic regression model by pooling comments and user ratings data across past articles. The global model is used to initialize the parameters of the per-article model. When training data is insufficient, the per-article model parameters shrink to the global parameters. Hence, the global model provides a good backoff estimate that helps to improve predictive accuracy for new articles. The global model is updated periodically (every few days).

Per-Article Rating Model: For an article a, a per-article logistic regression model is learnt at regular time intervals; at time t, our training procedure uses the features $\mathbf{X}$(t) and ratings $\mathbf{Y}(t)$ for comments in $C(t)$ as training data. The rationale for having a local per-article model is that once an article has a sufficient number of comments with ratings, the local model can provide better rating estimates than the global model for both existing as well as unseen comments. This is because it can capture article-specific feature-rating correlations, and deviations of the article from the average behavior captured by the global model – these deviations could be due to new article content never seen before, dynamically changing user sentiment, etc.

As mentioned earlier, when an article is new and has very little training data (cold-start scenario), our per-article model falls back to the global model and prediction accuracy is not adversely impacted. Furthermore, as the article collects more training data, the per-article model parameters start deviating to incorporate the article-specific comments and ratings. Finally, as each comment acquires more ratings, predictions become comment-specific based on the actual ratings the comment receives as opposed to its features. This is enabled by including a per-comment slack variable which helps to fill in the residuals not captured by the features. Thus, as the training data for a comment grows, our per-article model makes a smooth transition from feature-based predictions to comment-specific ones.

We should point out here that we don't include user features in our model because only a small fraction of users rate comments – as a result, (user, comment) ratings data is extremely sparse. Furthermore, in many cases, it is difficult to accurately identify users because they may not be logged in, or user cookie information is unreliable. Finally, even if we could identify users, certain information they may have provided in their profiles (e.g., age, salary, etc.) may be inaccurate, or they may have not been very active in terms of browsing or searching. In case reliable user features are available, these can be easily incorporated into our model to enable greater personalization.

Explore-Exploit: The explore-exploit module is responsible for selecting the K best comments to be displayed when a user views an article. Specifically, it uses the posterior estimates for the per-article model to obtain mean rating and variance estimates for each comment. These are subsequently combined to compute UCBs for each comment and the top-K comments with the highest UCB values are finally recommended to the user. Thus, LOGUCB exploits comments with high predicted ratings and explores the ones with high uncertainty in the predictions. The user rating feedback is incorporated into the training data for the article and used to further improve the per-article model.

Note that it is possible that the top-K comments with the highest UCBs may contain very similar or redundant comments – exploring/exploiting such comments can be wasteful. Recently, there has been some work [21, 25] on extending multi-armed bandit problems to select multiple arms per trial. However, the modeling assumption is that the K selected arms correspond to the K ranked results for a search query and at most one arm (corresponding to the first relevant result) gives a reward. Our comments scenario is somewhat different since ratings are determined by logistic regression and a user can give ratings to multiple comments in a single view. In this paper, we do not address the problem of diverse recommendations although LOGUCB can be used in conjunction with techniques such as *Maximal Marginal Relevance* (MMR) [11] (with UCB values substituted for relevance scores) to ensure comments diversity. Extending our comments recommendation framework to address diversity is left as future work.

The goal of bandit algorithms is to minimize the *regret*, that is, the difference between the expected reward of the algorithm and the optimal expected reward. In [16], LINUCB is shown to have an $\tilde{O}(\sqrt{T})$ regret over T trials for a fixed set of arms – this assumption, however, does not hold in our dynamic comments environment. Since logistic regression models are much more complex, formally proving regret bounds for LOGUCB is difficult and we do not focus on this in our work. However, logistic models are a natural choice (compared to linear models) for modeling binary ratings data, and in our experiments, we empirically show that our LOGUCB algorithm outperforms LINUCB.

In the following sections, we will provide a detailed description on each of the above three modules. For notational convenience, we will drop the argument t indicating time from $C(t)$, $\mathbf{Y}(t)$, etc. when it is clear from context.

4. GLOBAL MODEL

We start by describing our global probabilistic model and training procedure. We denote the comments (belonging to older articles) used for training the global model by C^g, and the corresponding features and ratings for the comments by $\mathbf{X}^g$ and $\mathbf{Y}^g$, respectively.

4.1 Model Specification

Since our ratings y_{ij} are binary, we assume that they come from a Bernoulli distribution with probability equal to the mean rating μ_i. Further, we model μ_i as a logistic function of the linear projection of the comment feature vector $\mathbf{x}_i$. Hence,

$$y_{ij} \sim \text{Bernoulli}(\mu_i) \quad (1)$$

$$\mu_i = \sigma(\mathbf{w}^\top \mathbf{x}_i) = \frac{1}{1+\exp(-\mathbf{w}^\top \mathbf{x}_i)} \quad (2)$$

where $\mathbf{w}$ is an M-dimensional weight parameter that captures the preferences of an average user for the different content features, and $\sigma(\cdot)$ is the logistic sigmoid function.

Data Likelihood: Given our ratings generation model, the likelihood of observations $\mathbf{y}_i$ for comments i is given by:

$$P(\mathbf{Y}^g|\mathbf{w}) = \prod_{i \in C^g} P(\mathbf{y}_i|\mathbf{w}) = \prod_{i \in C^g} \mu_i^{n_i^+} (1-\mu_i)^{n_i^-} \quad (3)$$

Prior over $\mathbf{w}$: We also assume a standard prior Gaussian distribution with zero mean and κ^2 variance over the weight vector $\mathbf{w}$:

$$P(\mathbf{w}) = N(\mathbf{w}|\mathbf{0}, \kappa^2) = \frac{1}{\sqrt{2\pi}\kappa} \exp(-\frac{\mathbf{w}^\top \mathbf{w}}{2\kappa^2}) \quad (4)$$

Posterior Distribution of $\mathbf{w}$: Using Bayes rule, the posterior distribution of $\mathbf{w}$ can be written as the product of the likelihood (in Equation (3)) and prior (in Equation (4)):

$$P(\mathbf{w}|\mathbf{Y}^g) \propto P(\mathbf{Y}^g|\mathbf{w})P(\mathbf{w})$$

4.2 Model Training

The goal of our model training procedure is to find the mode, also known as the *maximum-a-posteriori* (MAP) estimate, of the posterior distribution of $\mathbf{w}$. To find the mode $\mathbf{w}^g_{\mathsf{map}}$, we minimize the negative log of the posterior, that is,

$$\mathbf{w}^g_{\mathsf{map}} = \arg\min_{\mathbf{w}} -\sum_{i \in C^g} (n_i^+ \log(\mu_i) + n_i^- \log(1-\mu_i)) + \frac{1}{2\kappa^2}\mathbf{w}^\top \mathbf{w}$$

We use the standard iterative conjugate gradient descent [18] algorithm to obtain $\mathbf{w}^g_{\mathsf{map}}$.

5. PER-ARTICLE MODEL

As discussed earlier in Section 3, we learn a per-article model specifically for article a with the comments C of the article as training data. The set C contains N comments with features and ratings $\mathbf{X}$ and $\mathbf{Y}$, respectively.

5.1 Model Specification

As in the global model, we assume that the rating y_{ij} for a comment i comes from a Bernoulli distribution and the mean rating μ_i is a logistic function. However, it is possible that the comment feature space and/or the logistic model are incapable of predicting the rating correctly even if sufficient observations are available. Hence, the mean rating estimate $\hat{\mu}_i$ has a bias which is independent of the training data size. Thus, in addition to $\mathbf{w}^\top \mathbf{x}_i$, we add a per-comment slack variable z_i to our model for μ_i.

$$\mu_i = \sigma(\mathbf{w}^\top \mathbf{x}_i + z_i) = \frac{1}{1+\exp(-(\mathbf{w}^\top \mathbf{x}_i + z_i))} \quad (5)$$

We will use $\mathbf{z}$ to denote the vector of slack variables, that is, $\mathbf{z} = [z_1, z_2, \ldots, z_N]^\top$.

Prior over $(\mathbf{w}, \mathbf{z})$: We assume a Gaussian prior over the weight vector $\mathbf{w}$ with mean equal to the MAP estimate, $\mathbf{w}^g_{\mathsf{map}}$, of our global model and variance κ_w^2. Thus, $P(\mathbf{w}) = N(\mathbf{w}|\mathbf{w}^g_{\mathsf{map}}, \kappa_w^2)$. Observe that this prior provides backoff estimates from the global model when an article is new and has very few comments.

We also place a zero-mean Gaussian prior on each z_i. Thus, $P(\mathbf{z}) = N(\mathbf{z}|\mathbf{0}, \kappa_z^2)$. Note that if a comment i has no ratings in the training data, then z_i is zero because of the prior and μ_i is predicted using the projection $\mathbf{w}^\top \mathbf{x}_i$ only. On the other hand, if the comment has many ratings, then data likelihood dominates the prior and we precisely estimate the bias z_i. Thus, our prediction reduces to the empirical rating estimate based only on the fraction of thumbs-ups.

Posterior Distribution of $(\mathbf{w}, \mathbf{z})$: Using Bayes rule, the posterior distribution of $(\mathbf{w}, \mathbf{z})$ can be written as

$$P((\mathbf{w}, \mathbf{z})|\mathbf{Y}) \propto P(\mathbf{Y}|\mathbf{w}, \mathbf{z})P(\mathbf{w})P(\mathbf{z})$$
$$\propto \prod_{i \in C} \mu_i^{n_i^+}(1-\mu_i)^{n_i^-} N(\mathbf{w}|\mathbf{w}^g_{\mathsf{map}}, \kappa_w^2) N(\mathbf{z}|\mathbf{0}, \kappa_z^2) \quad (6)$$

5.2 Model Training

Observe that the likelihood involves logistic functions and the priors are Gaussians. Hence, a closed form does not exist for the posterior distribution. A popular technique is to approximate the posterior of $(\mathbf{w}, \mathbf{z})$ by a Gaussian distribution using the Laplace approximation [19]. The mean of the Gaussian is approximated by the mode $(\mathbf{w}_{\mathsf{map}}, \mathbf{z}_{\mathsf{map}})$ of the true posterior, while the inverse of the covariance matrix $\mathbf{S}$ is approximated by the Hessian matrix which comprises the second order derivatives of the negative log likelihood at the mode. Thus, the posterior $P((\mathbf{w}, \mathbf{z})|\mathbf{Y})$ is approximated as $N((\mathbf{w}, \mathbf{z})|(\mathbf{w}_{\mathsf{map}}, \mathbf{z}_{\mathsf{map}}), \mathbf{S})$.

The mode $(\mathbf{w}_{\mathsf{map}}, \mathbf{z}_{\mathsf{map}})$ can be found by minimizing $L = -\log P((\mathbf{w}, \mathbf{z})|\mathbf{Y})$, the negative log of the posterior. Thus,

$$(\mathbf{w}_{\mathsf{map}}, \mathbf{z}_{\mathsf{map}}) = \arg\min_{(\mathbf{w}, \mathbf{z})} -\sum_{i \in C}(n_i^+ \log(\mu_i) + n_i^- \log(1-\mu_i)) + \frac{1}{2\kappa_w^2}(\mathbf{w} - \mathbf{w}^g_{\mathsf{map}})^\top (\mathbf{w} - \mathbf{w}^g_{\mathsf{map}}) + \frac{1}{2\kappa_z^2}\mathbf{z}^\top \mathbf{z}$$

We use the standard iterative conjugate gradient descent algorithm to minimize L where the gradients are given by

$$\frac{\partial L}{\partial \mathbf{w}} = -\sum_{i \in C}(n_i^+(1-\mu_i) - n_i^- \mu_i)\mathbf{x}_i + \frac{1}{\kappa_w^2}(\mathbf{w} - \mathbf{w}^g_{\mathsf{map}})$$

$$\frac{\partial L}{\partial \mathbf{z}} = -\sum_{i \in C}(n_i^+(1-\mu_i) - n_i^- \mu_i) + \frac{1}{\kappa_z^2}\mathbf{z}$$

Next, we derive the equations for the covariance matrix $\mathbf{S}$ which is obtained by inverting the $M+N$-dimensional Hessian matrix.

$$\mathbf{S}^{-1} = \frac{\partial^2 L}{\partial (\mathbf{w}, \mathbf{z})^2} \Big|_{(\mathbf{w}_{\mathsf{map}}, \mathbf{z}_{\mathsf{map}})} \quad (7)$$

Data: New article a, Global comments set C^g;
Variance for priors $\kappa, \kappa_w, \kappa_z$;
Explore-exploit parameter α;

begin
  Train global model to find MAP estimate $\mathbf{w}^g_{\mathsf{map}}$;
  Initialize comment set C for a to $\emptyset$;
  Initialize per-article model for a using priors for $(\mathbf{w}, \mathbf{z})$;
  for *each user viewing article* a **do**
    for *each comment* $i \in C$ **do**
      Estimate the expected rating $\hat{\mu}_i$ and its variance σ_i^2 using the per-article model;
      Set $\text{UCB}_i \longleftarrow \hat{\mu}_i + \alpha\sigma_i$;
    end
    Select top-K comments with the highest UCBs to show to user;
    Add new comments to C and update (features, ratings) information ($\mathbf{X}$, $\mathbf{Y}$) based on user feedback;
    Retrain per-article model if sufficient number of new ratings are added to $\mathbf{Y}$;
  end
end

Algorithm 1: The LOGUCB Algorithm.

where,

$$\frac{\partial^2 L}{\partial(\mathbf{w},\mathbf{z})^2} = \begin{pmatrix} \frac{\partial^2 L}{\partial \mathbf{w}^2} & \frac{\partial^2 L}{\partial \mathbf{w}\partial \mathbf{z}} \\ \frac{\partial^2 L}{\partial \mathbf{z}\partial \mathbf{w}} & \frac{\partial^2 L}{\partial \mathbf{z}^2} \end{pmatrix} = \begin{pmatrix} \frac{1}{\kappa_w^2}\mathbf{I}_{M\times M} + \mathbf{X}\mathbf{P}\mathbf{X}^\mathbf{T} & \mathbf{X}\mathbf{P} \\ \mathbf{P}\mathbf{X}^\mathbf{T} & \frac{1}{\kappa_z^2}\mathbf{I}_{N\times N} + \mathbf{P} \end{pmatrix}$$

Above, $\mathbf{P}$ is a $N \times N$ diagonal matrix, where $\mathbf{P}[i,i] = n_i\mu_i(1-\mu_i)$. The overall time complexity of computing the covariance matrix $\mathbf{S}$ is $O(M^2N + (M+N)^3)$. Here, $O((M+N)^3)$ is the time complexity of inverting the $M+N-$dimensional Hessian. If the number of dimensions $M+N$ is large, then we simply use the diagonal approximation of the Hessian. The complexity of finding the inverse then reduces to $O(M+N)$ and the overall time complexity for computing the covariance becomes $O(MN)$. Thus, model training can be achieved in a matter of a few seconds for thousands of dimensions and comments (see Section 7.5).

6. EXPLORE-EXPLOIT

Our LOGUCB algorithm (see Algorithm 1) adopts a UCB-based approach to handle the exploitation-exploration trade-off for our logistic regression rating prediction model. For each comment i, it uses Bayesian estimation methods to compute the mean rating estimate $\hat{\mu}_i$ and its variance σ_i^2. Specifically, $\hat{\mu}_i$ and σ_i^2 are chosen as the predictive mean and variance, respectively, of the mean rating μ_i over the posterior of the per-article model. Now, by Chebyshev's inequality, it follows that for an appropriate choice of α, $|\mu_i - \hat{\mu}_i| \leq \alpha\sigma_i$ holds with high probability ($\geq 1 - \frac{1}{\alpha^2}$). Thus, $\hat{\mu}_i + \alpha\sigma_i$ is a UCB on μ_i, and our algorithm selects the comments with the top-K UCB values to show to users. We collect the user feedback (thumbs-up/down ratings) on the displayed comments as well as any new comments posted by the user. Clearly, since only a small fraction of users typically rate comments, the estimated model parameters will not change (by much) often. Consequently, we keep collecting new observations until we have a sufficient number of new ratings – we then perform a periodic batch update of the per-article model parameters (as in [16]). Below, we present approximation techniques for computing the mean rating estimate $\hat{\mu}_i$ and its variance σ_i^2. The method for estimating $\hat{\mu}_i$ is from [19] while the approximation scheme for σ_i^2 is novel.

Mean Rating Prediction: Following the Bayesian approach, we predict the mean rating μ_i by its posterior predictive mean, that is, the expectation of μ_i with respect to the posterior of model parameters $(\mathbf{w}, \mathbf{z})$ conditional on observed data. Thus,

$$\hat{\mu}_i = \int \mu_i P((\mathbf{w},\mathbf{z})|\mathbf{Y}) d\mathbf{w} d\mathbf{z} \approx \int \sigma(\mathbf{w}^\top \mathbf{x}_i + z_i) N((\mathbf{w},\mathbf{z})|(\mathbf{w}_{\mathsf{map}}, \mathbf{z}_{\mathsf{map}}), \mathbf{S}) d\mathbf{w} d\mathbf{z}$$

Substituting $t = \mathbf{w}^\top \mathbf{x}_i + z_i$ in the above equation, we get

$$\hat{\mu}_i \approx \int \sigma(t) N(t|\mu_t, \vartheta_t^2) dt$$

where $\mu_t = \begin{bmatrix} \mathbf{w}_{\mathsf{map}} \\ \mathbf{z}_{\mathsf{map}} \end{bmatrix}^\top \begin{bmatrix} \mathbf{x}_i \\ \mathbf{u}_i \end{bmatrix}$ and $\vartheta_t^2 = \begin{bmatrix} \mathbf{x}_i \\ \mathbf{u}_i \end{bmatrix}^\top \mathbf{S} \begin{bmatrix} \mathbf{x}_i \\ \mathbf{u}_i \end{bmatrix}$ for $\mathbf{u}_i = [\mathbf{0}_{i-1}\ \mathbf{1}\ \mathbf{0}_{N-i}]^\top$.

Note that the above integral over t represents the convolution of a Gaussian with a logistic function, and cannot be solved analytically. However, as pointed out in [19], the logistic function $\sigma(t)$ can be approximated well by the scaled probit function $\phi(\lambda t)$, where $\phi(t) = \int_{-\infty}^{t} N(\theta|0,1)d\theta$. The parameter λ is found by making the slopes of the two functions same at the origin, which gives $\lambda^2 = \frac{\pi}{8}$. Now, the convolution of a probit with a Gaussian can be expressed in terms of another probit function. Specifically,

$$\begin{aligned} \hat{\mu}_i &\approx \int \phi(\lambda t) N(t|\mu_t, \vartheta_t^2) dt \\ &= \phi\left(\frac{\mu_t}{\sqrt{\lambda^{-2} + \vartheta_t^2}}\right) \\ &\approx \sigma\left(\frac{\mu_t}{\sqrt{1 + \frac{\pi\vartheta_t^2}{8}}}\right) \end{aligned} \quad (8)$$

The computational complexity to predict the mean rating is $O((M+N)^2)$ which is the number of steps needed to compute ϑ_t^2. However, if $\mathbf{S}$ is approximated by a diagonal matrix, then the time complexity reduces to $O(M+N)$. Note that the inverse computation needed to compute $\mathbf{S}$ is done only once (when the per-article model is trained) for all the comments.

Variance of Mean Rating Estimate: The variance σ_i^2 is given by the predictive variance of the mean rating μ_i which is

$$\begin{aligned} \sigma_i^2 &= E_{(\mathbf{w},\mathbf{z})}[\mu_i^2|\mathbf{Y}] - E_{(\mathbf{w},\mathbf{z})}[\mu_i|\mathbf{Y}]^2 \\ &= \int \sigma^2(\mathbf{w}^\top \mathbf{x}_i + z_i) N((\mathbf{w},\mathbf{z})|(\mathbf{w}_{\mathsf{map}}, \mathbf{z}_{\mathsf{map}}), \mathbf{S}) d\mathbf{w} d\mathbf{z} - \hat{\mu}_i^2 \\ &= \int \sigma^2(t) N(t|\mu_t, \vartheta_t^2) dt - \hat{\mu}_i^2 \end{aligned}$$

Above, the final equation is obtained by substituting $t = \mathbf{w}^\top \mathbf{x}_i + z_i$, and μ_t and ϑ_t are as defined earlier.

Again, the convolution of a Gaussian with a squared logistic function cannot be solved analytically. However, we observe empirically that a squared logistic function $\sigma^2(t)$ can be approximated well by a scaled and translated probit $\phi(\lambda(t+\gamma))$ (as opposed to only a scaled probit for the logistic function). We minimize the squared error between the two functions to empirically find the values of parameters λ and γ, which gives $\lambda = 0.703$ and

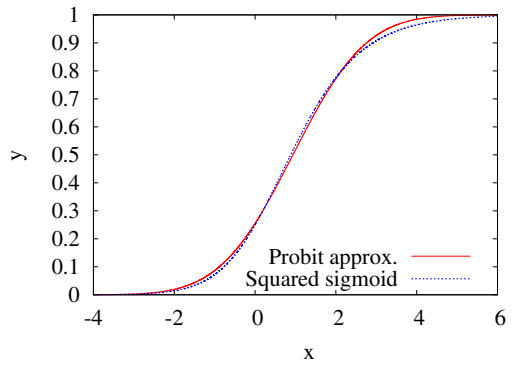


Figure 2: Probit approximation of σ^2.

$\gamma = -0.937$. The similarity between the squared logistic and probit functions for this choice of parameters is shown in Figure 2. The root mean square error is 0.0053 with a maximum error of 0.02. Again, since the convolution of a probit with a Gaussian can be expressed in terms of another probit function, we have,

$$\begin{aligned}\sigma_i^2 &\approx \int \phi(\lambda(t+\gamma))N(t|\mu_t, \vartheta_t^2)dt - \hat{\mu}_i^2 \\ &= \phi(\frac{\mu_t + \gamma}{\sqrt{(\lambda^{-2} + \vartheta_t^2)}}) - \hat{\mu}_i^2 \\ &\approx \sigma^2(\frac{\mu_t + \gamma}{\sqrt{1 + \vartheta_t^2\lambda^2}} - \gamma) - \hat{\mu}_i^2 \end{aligned} \quad (9)$$

Note that the variance σ_i^2 above is the difference between the squares of two sigmoids and is thus upper bounded by 1. However, such a bound does not hold for the variance of linear regression models. Thus, the UCB values for LOGUCB are more accurate compared to LINUCB [16].

We should also point out here that [23] presents an alternate variance estimation method for logistic regression which computes the variance of the Taylor approximation of μ_i (that ignores higher-order terms in the Taylor expansion of μ_i). However, since the probit approximates the squared logistic function so closely (see Figure 2), we expect our variance estimate to be more accurate.

7. EXPERIMENTAL RESULTS

In this section, we perform multiple experiments that demonstrate the effectiveness of our proposed methods on a real-life comments dataset from Yahoo! News. Our first experiment shows that our per-article logistic regression models incorporating global priors and leveraging bag-of-words and topic features have higher predictive accuracy compared to other models such as linear regression. In our second experiment, we show that our LOGUCB algorithm outperforms state-of-the-art explore-exploit methods. Finally, we report the running times of our per-article model training and rating prediction procedures.

7.1 Dataset

We obtain the comments rating data between March and June, 2011 from Yahoo! News. The ratings for different comments are available in terms of the number of thumbs-ups and thumbs-downs given by different users. Note that no ground truth per-comment mean rating μ_i is available. Hence, in order to reliably evaluate the performance of our algorithms, we only consider the comments that receive at least 10 ratings and assume that the ground truth mean rating μ_i is the same as the empirical mean given by the fraction of thumbs-ups. In our experiments, we consider 780 articles with 408,643 comments and a total of 16.8 million ratings.

We randomly select 70% of the articles and consider their comments as the global comments set C^g used to train the global model. Another 5% of the articles are used as a validation set to learn parameters of the different models. The remaining 198 articles are used to train per-article models for the model accuracy and explore-exploit experiments.

7.2 Features

We consider two different types of features.

Bag of words: The text of each comment is cleaned using standard techniques like lower-casing, removal of stop-words, etc. For the articles used to train the global model, we prune all words with very low (present in less than 10 comments) and high (present in greater than 50% of the comments) frequencies. The final vocabulary consists of approximately 9,000 words. On the other hand, for articles on which we train a per-article model, the low frequency threshold for pruning words is set to a lower value of 3 – this is because there are a lot fewer comments per article. We consider both L_2 normalized *tf* and *tf-idf* as feature values for words in each comment. The *tf* feature values give slightly better performance – so all the results are shown using them.

Topics: *Latent Dirichlet Allocation* (LDA) [10, 13] is a topic model that represents each document as a mixture of (latent) topics, where each topic is a probability distribution over words. LDA clusters co-occurring words into topics, and is able to overcome data sparsity issues by learning coarse-grained topic-rating correlations. We compute topic assignments to words in a comment using Gibbs sampling as described in [13], and use the topic distribution for a comment as feature values. The topic-word distributions are first learnt on the global comments set, and subsequently used to assign topics to the comments of each article. In our experiments, we set the default number of topics to 50.

We also considered a number of other per-comment features like (1) the fraction of positive and negative sentiment words where the sentiment of a word is obtained using Senti-WordNet[1], (2) number of special characters like question marks (?) or exclamations (!), and (3) total number of words, fraction of non-stop words, and fraction of upper-case words. However, none of these features had an effect on model performance.

7.3 Model Accuracy Experiments

In this experiment, we compare the mean rating prediction accuracy of different models.

7.3.1 Experimental Setup

Models Compared: We compare the following variants of our per-article model as well as other state-of-the-art models.

• *Global:* This is our global logistic regression model described in Section 4 trained on the comments set C^g. The features comprise the (unpruned) words in C^g and LDA topics.

• *PA1:* This is our per-article logistic regression model described in Section 5 with slack variables z_i set to zero and no global prior. It is trained on the article's comments, and features consist of the words in the article's comments and LDA topics.

• *PA2:* This is the same as *PA1* but uses the global prior and so has additional features corresponding to the words in C^g. We consider

[1] *http://sentiwordnet.isti.cnr.it/.*

Baselines	Training Size (%)				
	5%	10%	25%	50%	75%
Global	0.68	0.68	0.68	0.68	0.68
PA1	0.64	0.66	0.68	0.72	0.73
PA2	**0.68**	**0.69**	**0.72**	**0.74**	**0.76**
LinReg	0.64	0.66	0.68	0.72	0.73

Table 1: AUC scores for different models in Section 8.3.1

Training Size (%)	# Topics			
	0	25	50	100
5%	0.66	0.67	**0.68**	0.68
10%	0.68	0.68	**0.69**	0.69
25%	0.71	0.70	**0.72**	0.72

Table 2: AUC scores for $PA2$ for different number of topics.

this model to measure the impact of the global prior on prediction accuracy.

• *LinReg:* This is a linear model, $\mu_i = \mathbf{w}^\top \mathbf{x}_i$, to predict the mean rating scores. As in [16], we use ridge regression to solve for the coefficients $\mathbf{w}$. We choose a simple linear model since this is the underlying model for the state-of-the-art feature-based LINUCB explore-exploit algorithm [16]. The model is trained on an article's comments with the comment words and LDA topics as features.

Note that we do not consider models with slack variables since they mainly impact the rating estimates of training comments and not test comments. However, as shown later, slack variables do lead to performance improvements in an explore-exploit setting where there is no separate notion of test data. The variance parameters κ for the global model, and κ_w and κ_z for the per-article models are tuned using the validation set.

Evaluation Methodology: For model comparisons, the (non-global) comments in each of the 198 articles are further split into training comments (75%) and test comments (25%). The training comments for each article are used to train the per-article models *PA1*, *PA2*, and *LinReg*, and the test comments are used to measure the accuracy of all 4 models.

Evaluation Metric: Since we are looking to recommend comments with high mean rating scores to users, we are primarily concerned about the rating prediction accuracy for comments with extreme mean rating values. Furthermore, we want the predicted ratings for comments at the two extremes to be well separated. We use the *Area Under the Receiver Operating Characteristic (ROC) Curve* (AUC) metric to evaluate the predictive performance of the various models.

AUC is a popular measure of classifier accuracy on data with binary labels. We only choose test comments i with extreme mean ratings $\mu_i < 0.25$ and $\mu_i > 0.75$. Further, we assign positive labels to test comments with $\mu_i > 0.75$ and negative labels to comments with $\mu_i < 0.25$. The AUC is then computed based on the correctly and incorrectly classified comments for different classification thresholds on the predicted mean rating. We finally report the average AUC over all the articles. Note that a higher AUC value implies better predictive performance. Also, a random or constant rating prediction yields an AUC of 0.5 irrespective of the number of positive and negative examples in the test set.

7.3.2 Experimental Results

Our results show that both the global prior and per-article model are critical for achieving high prediction accuracy. Table 1 shows the AUC scores for the models discussed in Section 7.3.1 for different training set sizes. First, note that the global model (*Global*) has an AUC score greater than 0.5 and so significantly outperforms a model that makes random predictions. This clearly shows that a global model learned by pooling the articles together is useful. Furthermore, the two per-article models (*PA1, PA2*) provide better rating estimates than *Global* when sufficient training data is available. This is because per-article models are able to effectively capture article-specific feature-rating correlations involving new article content. However, since *PA1* does not use the global prior, it overfits and performs worse than *Global* when the article is new and hence training data is sparse. Introducing the global prior in the per-article model *PA2* ensures better predictive performance than both *Global* and *PA1* at all the training set sizes. Note that the AUC scores of the linear *LinReg* model are similar to those of *PA1* but lower than *PA2* because of the global prior. However, as we show later, *PA1* (and *PA2*) outperforms *LinReg* in an explore-exploit setting due to better variance estimates.

Table 2 shows the AUC scores of the *PA2* model for different number of LDA topics and training set sizes. In general, the bag-of-words feature (corresponding to 0 topics) turns out to be the most important for our models. Topic features help to improve the performance of our models when training sets are small. This is because topics are essentially clusters of words that frequently co-occur, and using them as features allows us to overcome data sparsity issues by generalizing correlations between a few topic words and ratings to the other words within the topic. For large training set sizes and number of topics beyond 50, however, we didn't see an improvement in AUC scores on adding topic features. We should point out here that all the observations made above (Tables 1 and 2) are statistically significant using the one-sided paired t-test with significance level 0.01.

7.4 Explore-Exploit Experiments

In this experiment, we compare the total expected number of thumbs-up ratings for different explore-exploit algorithms.

7.4.1 Experimental Setup

Explore-Exploit Schemes Compared: We consider the following context-free (without features) as well as contextual (with features) bandit schemes for evaluation.

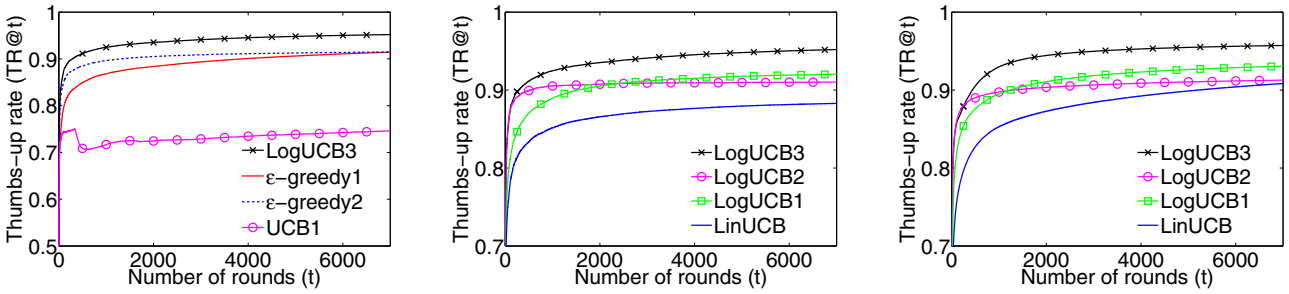
• *Random:* This policy chooses the top-K comments randomly.

• *UCB1:* This is a context-free UCB-based policy from [6]. In round t, the mean rating μ_i for a comment i is estimated as the current fraction of thumbs-ups the comment has received so far. The confidence interval for the estimate is given by $\sqrt{\frac{2\log t}{n_i(t)}}$, where $n_i(t)$ is the total number of ratings received by the comment until round t.

• *ϵ-greedy1:* This context-free policy chooses a random comment to show with probability ϵ, and with probability $1-\epsilon$, chooses the comment with the highest current fraction of thumbs-ups. To obtain the top-K comments, we repeat the process K times (each time excluding the comments that have already been selected).

• *ϵ-greedy2:* This is the contextual version of *ϵ-greedy* discussed above. With probability $1-\epsilon$, it chooses the comment with the highest predicted mean rating given by the per-article logistic regression model with global prior and slack variables (see Equation (8)).

• *LinUCB:* This algorithm represents the state-of-the-art and uses the *LinReg* model to obtain the mean rating estimate $\hat{\mu}_i$. Similar to



(a) $K = 1$ (b) $K = 1$ (c) $K = 10$

Figure 3: Performance of different explore-exploit algorithms.

our formulation, in [16], a closed form confidence bound is computed based on the predictive variance of the mean rating.

• *LogUCB1:* This is our LOGUCB algorithm (Algorithm 1) with per-article model *PA1* (without global prior) used for rating prediction.

• *LogUCB2:* This is our LOGUCB algorithm with per-article model *PA2* (with slack variables set to 0) used for rating prediction.

• *LogUCB3:* This is our full LOGUCB algorithm (Algorithm 1) with non-zero slack variables.

The parameters ϵ for ϵ-*greedy1* and ϵ-*greedy2*, and α for *LinUCB* and *LogUCB* variants are tuned using the validation set. Also, since *PA2* has a large number of features, we use the diagonal approximation of the Hessian in *LogUCB2* and *LogUCB3* for computing the covariance (see Section 5.2).

Evaluation Methodology: For evaluating the different explore-exploit schemes, the train-test split is not needed and we use all the comments of the 198 articles. Further, we evaluate performance using an offline simulation over 7000 rounds with new comment arrivals for an article following a poisson distribution. For a given explore-exploit algorithm, in each round t, the simulator takes the top-K comments chosen by the algorithm. For each of the K comments, it generates a binary thumbs-up/down rating from a Bernoulli distribution with probability given by the ground truth mean rating of the comment. These ratings are then provided to the algorithm as feedback, and the underlying models and mean rating estimates are appropriately updated at the end of the round.

Evaluation Metrics: We define the *Thumbs-up rate at round* t (TR@t) as the fraction of thumbs-up ratings received until round t. We use TR@t to evaluate the different explore-exploit algorithms since it is a good proxy for the total expected number of thumbs-up ratings that the algorithms are trying to maximize.

7.4.2 Experimental Results

Figure 3 plots TR@t values for the different explore-exploit algorithms discussed in Section 7.4.1 as a function of the number of rounds t.

Figure 3(a) compares the different context-free schemes with our logistic regression-based schemes for $K = 1$. Note that $K = 1$ corresponds to the standard single-slot multi-armed bandit problem. We do not present the curve for *Random* since it performs significantly worse than the other schemes with TR@t values below 0.65. Observe that ϵ-*greedy2* performs significantly better than ϵ-*greedy1* and *UCB1* when the number of rounds is small. This clearly shows that using the predictions of our feature-based logistic model in the exploit phase is more useful than empirical estimates of ratings in either ϵ-*greedy1* or *UCB1*. When the number of rounds become large and empirical estimates of ratings become reliable, the performance of ϵ-*greedy1* and ϵ-*greedy2* are comparable. Also, note that *LogUCB3* consistently outperforms ϵ-*greedy2*. Thus, the UCB value which includes model variance in *LogUCB3* is a better exploration strategy than the random one in ϵ-*greedy2*.

Figures 3(b) and 3(c) compare the different variants of our LOGUCB algorithm and LINUCB for $K = 1$ and $K = 10$, respectively. Observe that *LogUCB3* has a higher thumbs-up rate than both *LogUCB1* and *LogUCB2* – this can be attributed to more accurate rating estimates due to the use of slack variables in *LogUCB3*. Initially, when there are very few user ratings, *LogUCB2* does better than *LogUCB1* due to the global prior. However, the diagonal approximation for computing covariance causes its thumbs-up rate to fall below that of *LogUCB1* as the rounds progress. Finally, *LogUCB3* and the other *LogUCB* variants outperform *LinUCB* since as mentioned earlier, logistic regression has bounded mean rating and variance estimates (between 0 and 1), and is thus better suited for modeling binary ratings data. Again, it is worth mentioning here that the difference in performance of the various schemes in Figure 3 was found to be statistically significant with significance level 0.01.

# Comments	25	50	100	250	500	1000
Time (seconds)	0.87	0.92	1.0	1.13	1.28	1.63

Table 3: Training time for per-arcticle model $PA2$.

7.5 Running Time Experiments

Table 3 depicts the total time (in seconds) taken by the *LogUCB3* algorithm to train the per-article model and compute UCB values as the number of comments is increased from 25 to 1000. The running times are measured on a machine with 2 × 2.5 GHz Xeon processors and 16 GB of RAM.

In our experiments, we found that more than 90% of the time is spent on computing the MAP estimates $\mathbf{w}_{\mathsf{map}}$ and $\mathbf{z}_{\mathsf{map}}$ (see Section 5.2). Observe that model training and UCB computation are extremely fast and take only 1.63 seconds for 1000 comments. This is because we use the previous MAP estimates as the starting point in our conjugate gradient descent algorithm and the number of new ratings between two successive model updates is generally small – this causes the algorithm to converge within a few iterations. Moreover, gradient computations are also quick because comment feature vectors are sparse (comments are short with very few words). This is also the reason why computing the covariance (see Sec-

tion 5.2) and the UCB values (mean ratings in Equation (8) and variance in Equation (9)) takes only 26 and 148 msec, respectively, for all 1000 comments. Thus, our per-article models can be trained at frequent time intervals to give accurate rating estimates. We should also point out here that compared to 1.63 seconds taken by *LogUCB3*, *LinUCB* takes 0.95 sec to train models and compute UCB values for 1000 comments.

8. RELATED WORK

Recommender Systems: Recommender systems have been studied extensively in the literature. Popular methods predict user-item ratings based on neighborhoods defined using similarities between users/items [26], latent factors found using user-item matrix factorization [22], and generalizations of matrix factorization that model latent factors using regression on user/item features [1, 4] or item LDA topics [2]. However, these methods focus only on estimating ratings, and not on balancing the exploitation-exploration tradeoff which is so critical in our setting due to the dynamic nature of comments. Furthermore, due to fewer comment ratings per article, the feature weight and slack variable parameters in our per-article models both contribute to variance in the mean rating estimate – so we place Gaussian priors on both parameters (instead of only slack variables as is done in [1, 2]). Note that although our modeling framework currently models an average user, it can be easily extended to make personalized recommendations if user features are available.

Explore-Exploit: There exists a large body of work on the multi-armed bandit problem [12, 6, 15, 5, 16, 7]. Context-free bandit algorithms [6, 15, 5] such as ϵ-greedy and UCB assume that no side information is available and that arms are independent of each other. Such algorithms have been shown to achieve the lower bound of $O(\ln T)$ on the regret [15] (here, T is the number of trials). [3] develops a Bayesian solution and extends several existing bandit schemes to a dynamic set of items with short lifetimes, delayed feedback, and non-stationary reward distributions. However, the above methods assume that arms are independent, and hence do not model the correlation between the comments of a given article.

Contextual bandit algorithms [16, 7] assume that side information (e.g., features) is also available. LINREL [7] and LINUCB [16] are UCB-based algorithms that assume the reward of an arm is linearly dependent on its features. The algorithms have been shown to have an $\tilde{O}(\sqrt{T})$ regret for a fixed set of arms; however, in our problem setting, arms are dynamic. Furthermore, linear regression is not suitable for modeling binary ratings data, and cannot ensure that the average rating lies in the range $[0, 1]$.

Pandey et. al [20] assume that dependencies among arms can be described by a generative model on clusters of arms. However, these clusters are assumed to be known beforehand and each arm belongs to exactly one cluster. In contrast, our framework does not assume any clustering of comments. Instead, the dependencies among comments are effectively captured by our logistic regression model with comment features.

User Generated Content: Recently, analysis of user generated content has been an active area of research [24, 17, 28, 27, 11, 4, 14]. [11] uses different user, textual, and network features to analyze the credibility of tweets on twitter. [27] extends LDA [10] to model the generation of blog posts, authorship, as well as comments on the posts. [28] proposes a variant of supervised LDA [9] called the Topic-Poisson model to identify which blog posts will receive a high volume of comments. However, these methods do not focus on recommending comments to users. [14] uses an SVM regressor on different features such as unigrams, review length etc. to determine how helpful a review is. The helpfulness is defined as the fraction of users who found the review to be helpful. [24] analyzes the dependencies between comments, their ratings, and different topic categories. An SVM classifier is used to estimate the ratings of new or unrated comments. However, none of these methods employ an explore-exploit strategy, and are thus inadequate for ranking dynamic comments.

9. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel UCB-based explore-exploit algorithm called LOGUCB for recommending comments to users. LOGUCB uses a logistic regression model on word and topic features to predict the average rating for a comment. At its core, LOGUCB relies on a novel variance approximation technique under a Bayesian setting to derive the upper confidence bounds used to select comments. In experiments with a real-life comments dataset from Yahoo! News, LOGUCB outperforms other state-of-the-art context-free and feature-based explore-exploit methods like LINUCB. Directions for future work include extending our LOGUCB algorithm to ensure diversity of the K recommended comments, and proving formal regret bounds for our LogUCB algorithm.

10. REFERENCES

[1] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *SIGKDD*, pages 19–28, 2009.

[2] Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *WSDM*, pages 91–100, 2010.

[3] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. Explore/exploit schemes for web content optimization. In *ICDM*, 2009.

[4] Deepak Agarwal, Bee-Chung Chen, and Bo Pang. Personalized recommendation of user comments via factor models. In *EMNLP*, pages 571–582, July 2011.

[5] R. Agrawal. Sample mean based index policies with o(log n) regret for the multi-armed bandit problem. *Advances in Applied Probability*, pages 1054–1078, 1995.

[6] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47:235–256, May 2002.

[7] Peter Auer and M. Long. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3:2002, 2002.

[8] Chris Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[9] David Blei and Jon McAuliffe. Supervised topic models. In *NIPS*, 2008.

[10] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *JMLR*, 3:2003, 2003.

[11] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. Information credibility on twitter. In *WWW*, pages 675–684, 2011.

[12] J. C. Gittins. Bandit Processes and Dynamic Allocation Indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177, 1979.

[13] T. L. Griffiths and M. Steyvers. Finding scientific topics. *PNAS*, 101(Suppl. 1):5228–5235, April 2004.

[14] Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *EMNLP*, pages 423–430, 2006.

[15] Tze L. Lai and Herbert Robbins. Asymptotically efficient

adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985.

[16] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, pages 661–670, 2010.

[17] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *WWW*, pages 131–140, 2009.

[18] David Luenberger and Yinyu Ye. *Linear and Nonlinear programming*. Springer, 2008.

[19] D J C MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5), 1992.

[20] Sandeep Pandey, Deepayan Chakrabarti, and Deepak Agarwal. Multi-armed bandit problems with dependent arms. In *ICML*, 2007.

[21] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *ICML*, pages 784–791, New York, NY, USA, 2008.

[22] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887, 2008.

[23] Andrew Schein and Lyle Ungar. Active learning for logistic regression: An evaluation. *Machine Learning*, 69(3), 2007.

[24] Stefan Siersdorfer, Sergiu Chelaru, Wolfgang Nejdl, and Jose San Pedro. How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In *WWW*, pages 891–900, 2010.

[25] Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Learning optimally diverse rankings over large document collections. In *ICML*, pages 983–990. Omnipress, 2010.

[26] Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*, pages 501–508, 2006.

[27] Tae. Yano, William. Cohen, and Noah A. Smith. Predicting response to political blog posts with topic models. In *NAACL-HLT*, 2009.

[28] Tae. Yano and Noah A. Smith. What's worthy of comment? content and comment volume in political blogs with topic models. In *ICWSM*, 2010.