DIGTOBI: A Recommendation System for Digg Articles using Probabilistic Modeling

Younghoon Kim
Seoul National University
Seoul, Korea
yhkim@kdd.snu.ac.kr

Yoonjae Park
Seoul National University
Seoul, Korea
yjpark@kdd.snu.ac.kr

Kyuseok Shim †
Seoul National University
Seoul, Korea
shim@ee.snu.ac.kr

ABSTRACT

Digg is a social news website that lets people submit articles to share their favorite web pages (e.g. blog postings or news articles) and vote the articles posted by others. Digg service currently lists the articles in the front page by popularity without considering each user's preference to the topics in the articles. Helping users to find the most interesting Digg articles tailored to each user's own interests will be very useful, but it is not an easy task to classify the articles according to their topics in order to recommend the articles differently to each user.

In this paper, we propose DIGTOBI, a personalized recommendation system for Digg articles using a novel probabilistic modeling. Our model considers the relevant articles with low Digg scores important as well. We show that our model can handle both warm-start and cold-start scenarios seamlessly through a single model. We next propose an EM algorithm to learn the parameters of our probabilistic model. Our performance study with Digg data confirms the effectiveness of DIGTOBI compared to the traditional recommendations algorithms.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; H.3.5 [**Online Information Service**]: Web-based services

General Terms

Algorithm

Keywords

Digg article recommendation; Topic modeling; Collaborative filtering; Expectation-Maximization; Probabilistic latent semantic indexing

1. INTRODUCTION

Digg [10] is a social news website that allows people to submit articles for sharing their favorite web pages (e.g. blogs or news articles) and to vote the articles posted by others. When a Digg user finds an interesting web page with which he wants to share, he can submit an article to Digg so that other users can read his article and vote either thumbs up (also called *digging*) or thumbs down (also called *burying*) for the article. Each article in Digg consists of a user id, its submission title, a brief description and the link to the interesting web page as shown in Figure 1.

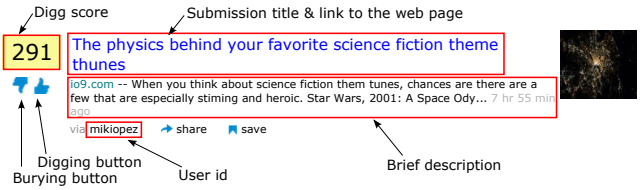


Figure 1: Components in a Digg submission

Digg service displays not only the submitted articles with a lot of "diggings" but also new submissions with their Digg scores where the Digg score of a submission is defined as the number of diggings subtracted by the number of buryings. Digg service currently lists the Digg articles in its front page without considering each user's topic preference. Helping users to find the most interesting Digg articles tailored to each user's own interests is very useful, but it is not an easy task to classify the submissions according to their topics so that the submitted articles are recommended differently to each user based on his own topic preference. Another challenge in Digg service is the problem of cold-start recommendations [21, 25] which occurs when the voting histories and submitted articles of users are not sufficient.

Other popular websites such as Facebook [11] and CiteULike [6] not only allow users to vote for the messages and the papers posted by other users respectively, but also post the voting scores which can be utilized to infer the topic preferences of users. Thus, our work in this paper can also be used to enhance the quality of recommendation systems for Facebook and CiteULike as well.

In our paper, we propose DIGTOBI which is a personalized recommendation system for DIGg arTicles using prObaBIlistic modeling. Our probabilistic model is a generalization of the probabilistic latent semantic indexing (PLSI) in [12] which assumes that the description in each submission has its own topic relevance model and each word in the description is selected by following the word distributions of its related topics. Since recommendation algorithms can be naturally derived in a principled manner by utilizing probabilistic modeling, it is another reason why we decide to apply the probabilistic modeling approach to the Digg article recommendation.

We develop a generative model from a unifying viewpoint such that the description in each submission is produced by

† Corresponding author.

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

a topic mixture model and each user votes thumbs up for each submission based on his topic preference. Since Digg service provides us only the dug articles and does not provide the buried articles by users, we could not utilize the votes of thumbs down in our model. If we blindly apply the PLSI model to Digg, we cannot fully utilize each user's digging history and submitted articles together. Thus, we introduce not only a topic mixture model of generating the descriptions in the submitted articles, as the PLSI model does, but also another topic model of producing the histories of digging and writing by each user, and apply both topic models into our probabilistic model. In our probabilistic model, we assume that every user writes the descriptions in his submissions according to his topic preference. Furthermore, we assume that when a user votes for submissions, the submissions with the similar topic relevances to his own topic preference have high chances of being voted for.

The traditional recommendation algorithms tend to offer users the articles with high scores which are probably preferred by most of users. However, users are interested in the relevant articles with low scores to their topic preferences as well. Thus, our model also considers the relevant articles with low Digg scores to be important. It is possible that the articles with high Digg scores may be dug by many users simply because they have more chances to be displayed in Digg service, and the others have no chance of being voted for at all. In our model, when an article with a small Digg score is dug by a user, we assume that the article has more similar topic relevance to the user's topic preference. This assumption enables our model to capture the topic preference of each user precisely and discover the interesting articles to each different user, even if the articles have low Digg scores.

The contributions of this paper are as follows:

- We develop the personalized recommendation system called DIGTOBI which utilizes our novel probabilistic generative model. The model is suitable for representing the activities in Digg service as well as the other websites which allow users to vote the contents posted by other users.
- We improve the quality of recommendations by reducing the bias of the collaborative filtering which mainly recommends the popular articles with high Digg scores.
- We present the EM algorithm which estimates the model parameters maximizing the likelihood of our probabilistic model used by DIGTOBI.
- We also introduce the recommendation algorithms for both warm-start and cold-start naturally derived from our probabilistic model of DIGTOBI in a principled manner.
- Our performance study with Digg data confirms the effectiveness of DIGTOBI compared to the traditional recommendation algorithms.

The rest of this paper is organized as follows. After discussing related work in Section 2, we provide the definitions to define our problem and present the problem formulation in Section 3. We next propose a generative probabilistic model and its EM algorithm in Section 4. Then, we present our recommendation algorithms in Section 5. Finally, the performance study is provided in Section 6 and we summarize our paper in Section 7.

2. RELATED WORK

Model-based algorithms build their models based on the behaviors of users and utilize the models to predict the users' preferences on unseen items. In [22], it was shown that recommendations using probabilistic modeling outperform other approaches in terms of precision. The probabilistic matrix factorization technique in [23] was also developed for movie recommendations in Netflix [20] to predict user ratings on movies. However, these algorithms utilize only the history of ratings or votings for the items generated by users and do not consider additional hints such as the textual content of the items for recommendations.

More complex probabilistic models were later proposed in various recommendation applications. In [13, 17, 26], recommendations utilizing the PLSI in [12] were investigated. The LDA model, introduced in [4], is generalized to model both latent topics and hidden communities between users in [1, 2]. However, these algorithms make recommendations by utilizing either the user's past ratings on items or the textual content of items, but not both.

In [28], a system called CTR was proposed to recommend scientific papers to each user in CiteULike [6]. CiteULike is a specialized search engine for searching the technical papers and allows people to share their favorite papers by voting thumbs up for the papers. To make recommendations by utilizing both of user's votes and the content of papers, they combine the *matrix factorization* [23] and the LDA model [4] together, and show that CTR outperforms the recommendations based on either matrix factorization or the LDA model only [28].

While both CiteULike and Digg service allow the users to post their votes for each document, CiteULike does not allow the users to post the descriptions of why they like. Digg service not only allows the users to submit the descriptions of web documents, but also posts the Digg scores. Thus, our probabilistic model is more general in that we consider both Digg scores and user descriptions, while CTR in [28] does not considered both of them. If we want to use the CTR model for recommending Digg articles, we can extend it by ignoring the Digg scores and the descriptions posted by users.

In [5], recommending URLs in Twitter messages was studied by exploiting the social network of users and the popularity of the URLs in Twitter. They simply represent Twitter messages of a user as a bag-of-word for content-based recommendations without considering any sophisticate user modeling, while our work utilizes probabilistic modeling.

To recommend the articles with hot topics in Digg service, HotDigg which is a probabilistic model based recommendation system was proposed recently in [14]. However, HotDigg aimed to recommend the articles on recent hot topics to general users using the Digg scores and the descriptions of submissions only, while we aims to provide personalized recommendations in this paper.

3. PRELIMINARIES

We first provide the definitions used for defining our recommendation problem and next present the problem formulation.

3.1 Problem Formulation

Let $D = \{d_1, ..., d_n\}$ be a collection of the Digg article ids submitted to Digg service. Let $U = \{u_1, ..., u_m\}$ be a set of

User	$L(u)$	$D(u)$	$W(d_i)$	s_i
u_1	d_5	d_1	Sherlock, Holmes, 221B, Baker	3
		d_2	iPhone, Apple, Samsung	246
u_2		d_3	iPhone, AppStore, Samsung	324
		d_4	Samsung, iPhone, AppStore, Baker	2
		d_5	Sherlock, 221B, Apple, Baker	3
u_3	d_3,d_4	d_6	Apple, iPhone, Sherlock	2
u_4	d_1,d_2,d_3	d_7	Sherlock, Holmes, iPhone	2

(a) Warm-start data set

User	$L(u)$	$D(u)$	$W(d_i)$	s_i	$U(d_i)$
u_5	d_2,d_3,d_4				
u_6		d_8	iPhone, AppStore, Samsung	1	
		d_9	Baker, Apple	1	u_2,u_3

(b) Cold-start data set

Figure 2: An example of users and articles

Digg user ids. A user can submit an article which consists of the user id as well as the title, a brief description and the link to the interesting web page of the submitted article. Let $D(u_a)$ denote the set of Digg article ids which are submitted by the user $u_a \in U$ and let $W(d_i)$ denote the bag-of-words appearing in either the title or the description of the article $d_i \in D$. The number of words in $W(d_i)$ is denoted by N_i and we use w_{ij} $(1 \leq j \leq N_i)$ to represent the j-th word appearing in the article $d_i \in D$. For each article $d_i \in D$, we generate the bag-of-words $W(d_i)$ by deleting the stop-words from its original title and description. Let $W = \{w_1, ..., w_\ell\}$ be the vocabulary which is the set of distinct words occurring in at least an article $d_i \in D$. We define $n(d_i, w)$ to denote the number of occurrences of the word $w \in W$ in $W(d_i)$.

Digg users can vote thumbs up or thumbs down, called *digging* or *burying* respectively, for each Digg article. Every article $d_i \in D$ has a *Digg score* s_i of an integer which is the number of diggings subtracted by the number of buryings voted for d_i. Let $L(u_a)$ denote the set of article ids in D for which the user $u_a \in U$ votes thumbs up. For burying, we cannot obtain the list of articles for which users vote thumbs down and thus we utilize the history of diggings only. Let $U(d_i)$ denote the set of users in U who dug the article $d_i \in D$. Finally, we refer to a user $u_a \in U$ to whom Digg articles are recommended as an *active user*.

Problem definition: Assume that we are given a collection D of Digg articles with Digg scores and a set of users U with the history of diggings for each user in U. The *top-K Digg article recommendation* problem is defined as follows:

PROBLEM 1. *For an active user $u_a \in D$ and a set of candidate articles C which are not submitted nor dug by the user u_a yet, the problem is to find the top-K Digg articles which the user u_a would like the most among the candidate articles in C.*

EXAMPLE 3.1: *Consider a set of users U=$\{u_1,...,u_4\}$ and a set of Digg articles D=$\{d_1,...,d_7\}$ in Figure 2(a). The articles dug by each user are presented in the column of $L(u)$. We can see that there exist two topics of 'smartphone' and 'Sherlock Holmes'. The topic of 'smartphone' is much more popular than that of 'Sherlock Holmes' in the Digg articles. Thus, the articles on 'smartphone' usually obtain higher scores and have more chances to be listed in the front page of Digg service.*

Suppose that we recommend the top-1 article for the user u_3 who posted the article d_6 and voted thumbs up for d_3 and d_4 both of which are submitted by u_2. Considering the articles he dug and submitted, we can see that he obviously wants to read the articles about 'smartphone'. If we simply consider only the other articles submitted (or voted) by the user u_2, who submitted the articles that u_3 mainly dug, we would recommend the article d_5 which is posted by u_2. However, the user u_3 prefers the topic of 'smartphone' and it is better to recommend the article d_2 on 'smartphone' rather than the article d_5 which is about the topic of 'Shelock Holmes'.

Now, suppose that we want to recommend the top-1 article for the user u_4 who submitted an article d_7 which contains more words referring to 'Sherlock Holmes' than 'smartphone'. From the history of his diggings, we may think that he is interested in the articles on 'smartphone' since he dug d_2 and d_3. Due to their high scores, the articles on 'smartphone' generally have more chances to be displayed in the front page of Digg service and thus the reason why u_4 dug them is probably because he has actually seen the articles d_2 and d_3 in the front page. However, considering the contents of d_7 submitted by the user u_4, recommending the article d_5, which is related to the topic 'Sherlock Holmes', is also important to the user u_4. ∎

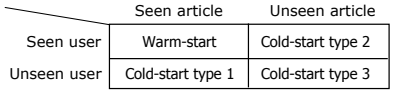


Figure 3: Four types of recommendations

3.2 Warm and Cold-start Recommendations

The top-K recommendation problem is classified into four categories based on whether the active users, the candidate articles to be recommended, or both of them are included in the training data, which is used to learn the parameters of models, as illustrated in Figure 3.

- **Warm-start recommendation:** This is the case when both of the active users and the candidate articles are *seen* (i.e., the active users and candidate articles were included in the training data used for learning the model).
- **Cold-start recommendation of type 1:** It is the recommendation of seen candidate articles to an *unseen* active user (i.e., the active user was not included in the training data).
- **Cold-start recommendation of type 2:** This refers to the recommendation of unseen candidate articles (i.e., the candidate articles were not included in the training data) to a seen active user.
- **Cold-start recommendation of type 3:** It is the case of recommending unseen candidate articles to an unseen active user.

4. OUR GENERATIVE MODEL FOR DIGG

4.1 Our Generative Model

Previous works have shown the effectiveness of topic mixture models in clustering text collections by representing hidden topics with conditional probability distributions[4, 12, 15, 18]. However, the traditional topic mixture models do not consider the ratings of text documents provided by users in order to get a clue for the topic preferences of users. To model both writing and digging behaviors of users, we use a model structure with the mixtures of conditional probability distributions by generalizing the PLSI model in [12] to consider the digging behaviors of users in Digg service.

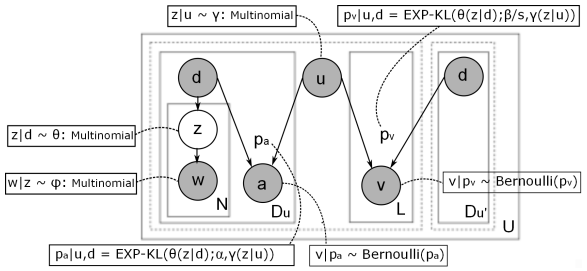


Figure 4: A graphical model

Observed data: We assume that our generative model produces the following observed data: (1) the bag-of-words $W(d_i)$ for every Digg article $d_i \in D$, (2) the list $D(u_a)$ of Digg article ids submitted by every user $u_a \in U$, (3) the list $L(u_a)$ of Digg article ids dug by every user $u_a \in U$.

Hidden topics: We assume that there exist t number of major topics denoted by the integers from $Z=\{1, 2, ..., t\}$ in a collection of Digg articles D. We let w_{ij} be the j-th word appearing in the article $d_i \in D$ and let a random variable z_{ij} be the *hidden variable* to represent one of the t number of topics according to which the word w_{ij} is chosen. Note that we do not know in advance the actual label with which we can identify the topic z_{ij}.

Conditional probability distribution functions (PDFs): We introduce the following three conditional PDFs to represent the topic relevance models of articles, topic preference models of users as well as word selection models of topics:

(1) **The topic relevance models of articles:** We introduce $\theta(z|d=d_i)$ which is the conditional PDF with which we select the topic $z \in Z$ for a given Digg article $d_i \in D$. It is a multinomial distribution over the topics in Z satisfying

$$\sum_{z_k \in Z} \theta(z=z_k|d=d_i) = 1 \quad \text{for every } d_i \in D, \tag{1}$$

(2) **The topic preference models of users:** We define $\gamma(z|u=u_a)$ that is the conditional PDF with which a user $u_a \in U$ chooses the topic $z \in Z$. It follows a multinomial distribution over the topics in Z and thus we have

$$\sum_{z_k \in Z} \gamma(z=z_k|u=u_a) = 1 \quad \text{for every } u_a \in U, \tag{2}$$

(3) **The word selection models in topics:** We use $\phi(w|z=z_k)$ to represent the conditional PDF from which the word $w \in W$ is drawn, given the topic $z_k \in Z$. It is also a multinomial distribution over the words in W which satisfies

$$\sum_{w_j \in W} \phi(w=w_j|z=z_k) = 1 \quad \text{for every } z_k \in Z \tag{3}$$

As long as it is clear from the context, we denote $\theta(z|d=d_i)$, $\gamma(z|u=u_a)$ and $\phi(w|z=z_k)$ by $\theta(z|d_i)$, $\gamma(z|u_a)$ and $\phi(w|z_k)$ respectively for the sake of simple representation.

Figure 4 illustrates the graphical representation of our proposed mixture model. Note that p_a and p_v, both of which will be dicussed in Section 4.2, denote the probabilities of submitting an article and digging an article by a user respectively. Our generative model includes two independent generative processes. The first process considers the behavior of writing Digg articles by users. The second one represents how Digg users dig the articles written by other users. Both processes are illustrated below:

Writing Digg articles: Each Digg article $d_i \in D(u_a)$ is produced by the user $u_a \in U$ with repeating the following steps while choosing the words stochastically:

- The user u_a decides to write the Digg article $d_i \in D(u_a)$ by following the Bernoulli distribution with the success probability computed by the function EXP-KL($\theta(z|d_i)$; $\alpha, \gamma(z|u_a)$) where α is a constant in our model. As the multinomial distributions $\theta(z|d_i)$ and $\gamma(z|u_a)$ become similar, the EXP-KL function generates high probability values. Similarly, as they become dissimilar, it produces low values closer to 0. We will discuss the EXP-KL function in the next section.
- Each word in $W(d_i)$ is selected by repeating the following two steps N_i times.
  1. To select the j-th word w_{ij} in d_i, we first choose a topic $z_{ij} \in Z$ by following the conditional PDF $\theta(z|d_i)$.
  2. With the topic z_{ij} selected in the above, the word w_{ij} is chosen according to the conditional PDF $\phi(w|z_{ij})$.

Digging Digg articles: A user u_a determines whether he digs each Digg article $d_i \in D(u_a)$ or not by following the Bernoulli distribution with the success probability of EXP-KL($\theta(z|d_i); \beta/s_i, \gamma(z|u_a)$) where s_i is the Digg score of d_i and β is a constant in our model. Since the Digg articles with high scores have more chances to be shown to each user, if the article d_i has a high Digg score s_i, the user u_a tends to dig d_i with a higher probability than the Digg articles with small Digg scores in the formulation of EXP-KL($\theta(z|d_i); \beta/s_i, \gamma(z|u_a)$).

We will next introduce the EXP-KL function used in our model and discuss its property.

4.2 The Exponential-KL Function

Given two multinomial distributions, we propose to utilize the *exponential KL-divergence (EXP-KL)* function in our probabilistic model in order to obtain higher success probabilities for Bernoulli distributions [30] as the two multinomial distributions become similar to each other. Given a scalar value λ and a t-dimensional multinomial distribution μ, the exponential KL-divergence function $EXP\text{-}KL(\theta; \lambda, \mu)$ over the t-dimensional multinomial distribution θ is defined as

$$EXP\text{-}KL(\theta; \lambda, \mu) = \lambda e^{-\lambda \cdot KL[\mu||\theta]}, \tag{4}$$

where $KL[\mu||\theta]$ is the Kullback-Leibler (KL) divergence between μ and θ, which is presented in [7] as

$$KL[\mu||\theta] = \sum_{i=1}^{t} \mu_i \cdot \log(\mu_i/\theta_i), \tag{5}$$

and λ is the parameter which determines the steepness of the EXP-KL function.

The EXP-KL function generates the probability density of λ when $\theta=\mu$. The probability densities calculated by using EXP-KL decrease with increasing the distance between θ and μ. Furthermore, with a large λ, the probability densities decrease faster when the distance between θ and μ grows. We next show how the EXP-KL function is utilized in our model in details.

The probability of submitting an article by a user: A user $u_a \in U$, whose topic preference is $\gamma(z|u_z)$, submits the article $d_i \in D$, of which topic relevance is $\theta(z|d_i)$, with the probability of

$$EXP\text{-}KL(\theta(z|d_i); \alpha, \gamma(z|u_a)) = \alpha e^{-\alpha \cdot KL[\gamma(z|u_a)||\theta(z|d_i)]}, \quad (6)$$

where α is a constant in our model. With a small α, it becomes more probable that an article with a quite different topic relevance of $\theta(z|d_i)$ from its author's topic preference is submitted. Thus, by setting a small value to α, we can allow Digg users to submit the articles with diverse topic relevances.

The probability of digging an article by a user: A user $u_a \in U$, whose topic preference is $\gamma(z|u_a)$, digs the article $d_i \in D$ with Digg score s_i, whose topic relevance is $\theta(z|d_i)$, with the probability of

$$EXP\text{-}KL(\theta(z|d_i); \beta/s_i, \gamma(z|u_a)) = \frac{\beta}{s_i} e^{-(\beta/s_i) \cdot KL[\gamma(z|u_a)||\theta(z|d_i)]}, \quad (7)$$

where β is a constant such that with a small value of β, it becomes more probable that a user u_a with the topic preference of $\gamma(z|u_a)$ votes thumbs up for the article d_i with a quite different topic relevance of $\theta(z|d_i)$ from his topic preference $\gamma(z|u_a)$. Furthermore, since the articles with high Digg scores not only are interesting to many users but also have more chances to be displayed in the front page of Digg, such articles will be dug by many users with higher probabilities. For the article d_i with the Digg score s_i, by using the steepness parameter (β/s_i) in the EXP-KL function, the users tend to dig the articles with larger Digg scores in our model.

4.3 The Likelihood of Digg Data

Let D and U be a set of Digg articles and a set of Digg users respectively. For each Digg article $d_i \in D$, we have a bag-of-words $W(d_i)$ and a Digg score s_i. For each Digg user $u_a \in U$, let $D(u_a)$ and $L(u_a)$ be the set of articles that the user u_a submitted and the set of articles that u_a dug respectively. Let $p_{post}(d_i|u_a)$ and $p_{digg}(d_i|u_a)$ represent the probability that u_a summits the Digg article d_i and the probability that u_a votes thumbs up for d_i respectively. Since each user $u_a \in U$ posts the Digg articles in $D(u_a)$ and digs those in $L(u_a)$ independently in our model, the likelihood $\mathbb{L}$ of the Digg data becomes

$$\mathbb{L} = \prod_{u_a \in U} \left\{ \left[\prod_{d_i \in D_u} p_{post}(d_i|u_a) \right] \cdot \left[\prod_{d_i \in L_u} p_{digg}(d_i|u_a) \right] \right\}.$$

Since each word in $W(d_i)$ is sampled independently after the user u_a decides to submit the article d_i, we have

$$p_{post}(d_i|u_a) = EXP\text{-}KL(\theta(z|d_i); \alpha, \gamma(z|u_a)) \cdot \prod_{w_{ij} \in W(d_i)} p(w_{ij}|d_i).$$

Furthermore, if we marginalize $p(w_{ij}|d_i)$ with the random variable z_{ij}, $p(w_{ij}|d_i)$ becomes $\sum_{z_k \in Z} \phi(w{=}w_{ij}|z_k)\, \theta(z{=}z_k|d_i)$. Finally, since each article in $L(u_a)$ is dug by u_a independently by following the distribution EXP-KL($\theta(z|d_i); \beta/s_i$, $\gamma(z|u_a)$), we obtain the likelihood as follows

$$\mathbb{L} = \prod_{u_a \in U} \prod_{d_i \in D(u_a)} \alpha e^{-\alpha \cdot KL[\gamma(z|u_a)||\theta(z|d_i)]} \cdot \prod_{u_a \in U} \prod_{d_i \in D(u_a)} \prod_{w_j \in W} \left[\sum_{z_k \in Z} \phi(w_j|z_k)\theta(z_k|d_i) \right]^{n(d_i, w_j)} \cdot \prod_{u_a \in U} \prod_{d_i \in L(u_a)} (\beta/s_i) e^{-(\beta/s_i) \cdot KL[\gamma(z|u_a)||\theta(z|d_i)]}, \quad (8)$$

where $n(d_i, w_j)$ denotes the number of appearances of the word $w_j \in W$ in the article d_i.

4.4 The Maximum Likelihood Estimate

Assume that the observed data is generated from our generative model. Let Θ denote our initially-unknown model parameters, which consist of the distributions $\phi(w|z_k)$ for every $z_k \in Z$, $\theta(z|d_i)$ for every $d_i \in D$ and $\gamma(z|u_a)$ for every $u_a \in U$. We wish to find the model parameters Θ such that the likelihood $\mathbb{L}$ in Equation (8) is maximized. This is known as the Maximum Likelihood (ML) estimation [19] for computing Θ. In order to estimate Θ, we generally introduce the log-likelihood function defined as

$$\begin{aligned} \log \mathbb{L} = & \sum_{u_a \in U} \sum_{d_i \in D(u_a)} \sum_{w_j \in W} n(d_i, w_j) \log \sum_{z_k \in Z} \phi(w_j|z_k)\theta(z_k|d_i) \\ & - \alpha \sum_{u_a \in U} \sum_{d_i \in D(u_a)} KL[\gamma(z|u_a)||\theta(z|d_i)] \\ & - \beta \sum_{u_a \in U} \sum_{d_i \in L(u_a)} (1/s_i) \cdot KL[\gamma(z|u_a)||\theta(z|d_i)] \\ & + \sum_{u_a \in U} \sum_{d_i \in D(u_a)} \log \alpha + \sum_{u_a \in U} \sum_{d_i \in L(u_a)} \log(\beta/s_i). \end{aligned} \quad (9)$$

The likelihood function is considered to be a function of the parameters Θ for the Digg data. Since $\log \mathbb{L}$ is a strictly increasing function, the parameters of Θ which maximize log-likelihood of $\log \mathbb{L}$ also maximize the likelihood $\mathbb{L}$ [31]. Note that the parameters $\theta(z|d)$, $\gamma(z|u)$ and $\phi(w|z)$ are probability values and thus we have the constraints of Equations (1)–(3). Using these constraints, we calculate the model parameters Θ with maximizing the log-likelihood $\log \mathbb{L}$ in Equation (9).

The effect of α and β: Note that KL-EXP($\gamma(z|u_z)||\theta(z|d_i)$) in the log-likelihood of Equation (9) can be represented as $-H[\gamma(z|u_a)] - \sum_z \log \theta(z|d_i)$ where $H[\gamma(z|u_a)]$ is the entropy of $\gamma(z|u_a)$. It is known that the entropy of multinomial distribution is maximized when the distribution is uniform [7]. Since KL-EXP($\gamma(z|u_z)||\theta(z|d_i)$) is multiplied by $-\alpha$ and $-\beta$ in Equation (9), with large α and β, maximizing the entropy terms increases the log-likelihood of Equation (9) more than maximizing the other terms in Equation (9) does. Thus, when α and β are large, the distribution of $\gamma(z|u_a)$ becomes closer to the uniform distribution. Furthermore, when both of α and β are zeros, our model becomes the original PLSI model in [12] which is a special case of our model.

4.5 Estimation of Model Parameters

Without any prior knowledge to the model parameters, we can apply the maximum likelihood estimator to compute all the parameters by applying the EM algorithm [8]. An

E-step:

$$p^{(k+1)}(z_{ij}{=}z_k|d_i,w_{ij}) = \frac{\phi^{(k)}(w{=}w_j|z_k)\theta^{(k)}(z{=}z_k|d_i)}{\sum_{z'\in Z}\phi^{(k)}(w{=}w_j|z')\theta^{(k)}(z{=}z'|d_i)} \tag{10}$$

M-step:

$$\phi^{(k+1)}(w{=}w_j|z_k) = \frac{\sum_{d_i\in D} n(d_i,w_j)\cdot p^{(k+1)}(z{=}z_k|d_i,w_j)}{\sum_{w'\in W}\sum_{d_i\in D} n(d_i,w')\cdot p^{(k+1)}(z{=}z_k|d_i,w')} \tag{11}$$

$$\theta^{(k+1)}(z{=}z_k|d_i) = \frac{\sum_{w_j\in W} n(d_i,w_j)\cdot p^{(k+1)}(z{=}z_k|d_i,w_j) + \alpha\gamma^{(k)}(z{=}z_k|u_a) + (\beta/s_i)\sum_{u'\in U(d_i)}\gamma^{(k)}(z{=}z_k|u')}{\sum_{z'\in Z}\{\sum_{w_j\in W} n(d_i,w_j)\cdot p^{(k+1)}(z{=}z'|d_i,w_j) + \alpha\gamma^{(k)}(z{=}z'|u_a) + (\beta/s_i)\sum_{u'\in U(d_i)}\gamma^{(k)}(z{=}z'|u')\}} \tag{12}$$

$$\gamma^{(k+1)}(z{=}z_k|u_a) = \frac{\left[\prod_{d_i\in D(u_a)}\theta^{(k+1)}(z{=}z_k|d_i)^{\alpha}\prod_{d_j\in L(u_a)}\theta^{(k+1)}(z{=}z_k|d_j)^{\beta/s_j}\right]^{1/(\alpha|D_u|+\sum_{d_j\in L(u_a)}\beta/s_j)}}{\sum_{z'\in Z}\left[\prod_{d_i\in D(u_a)}\theta^{(k+1)}(z{=}z'|d_i)^{\alpha}\prod_{d_j\in L(u_a)}\theta^{(k+1)}(z{=}z'|d_j)^{\beta/s_j}\right]^{1/(\alpha|D_u|+\sum_{d_j\in L(u_a)}\beta/s_j)}} \tag{13}$$

Figure 5: The formulas for E-step and M-step

EM algorithm performs the iterations with the two steps of an expectation step (E-step) and a maximization step (M-step). In the E-step, the probability distributions of the hidden variables are computed by using the current estimate of parameters, and in the M-step, the parameters maximizing the log-likelihood are calculated by utilizing the expectation computed in the E-step. The parameters estimated in the M-step are then used in the E-step of the next iteration.

The E-step: This step calculates the expectation of the hidden variables. Each hidden variable is the topic z_{ij} which is chosen for selecting the word w_{ij} (i.e., the j-th word occurring at d_i). Let $p(z_{ij}{=}z_k|d_i,w_{ij})$ be the probability that a word w_{ij} is generated from the topic z_k in the Digg article d_i. The formula to compute $p^{(k+1)}(z_{ij}{=}z_k|d_i,w_{ij})$ in the E-step of the $(k{+}1)$-th iteration using the model parameters $\Theta^{(k)}$ computed in the k-th iteration is presented in Figure 5.

The M-step: In order to find the parameters $\Theta^{(k+1)}$ maximizing Equation (9), we apply the method of Lagrange multipliers [3]. The obtained formulas for the M-Step to update the model parameters Θ at the $(k{+}1)$-th step are listed in Figure 5. Note that the topic preference $\gamma(z{=}z_k|u_a)$ of the user u_a is calculated in the form of geometric mean, which is commonly used to compute the average of ratio values [29], with the topic relevances $\theta(z{=}z_k|d_j)$ of the Digg articles written and dug by the user u_a.

We iterate the E-Step and M-Step until we obtain the convergence of the log-likelihood in Equation (9). Since our EM algorithm only guarantees to find a local maximum of the likelihood, we perform multiple trials and choose the best one among the local optima found.

5. RECOMMENDATIONS USING MODEL PARAMETERS

Once the parameters Θ in our model are estimated, recommendations can be made by utilizing the model parameters.

Top-K warm-start recommendation: To recommend the top-K Digg articles, for a user u_a and an article d_i, we predict the preference of the user u_a for an article d_i using the KL divergence between $\gamma(z|u_a)$ and $\theta(z|d_i)$ as

$$Score(u_a,d_i) = 1/KL[\gamma(z|u_a)||\theta(z|d_i)] \tag{14}$$

and recommend the top-K scored articles among the candidates. It is because the KL divergence between $\gamma(z|u_a)$ and $\theta(z|d_i)$ increases as both of $\gamma(z|u_a)$ and $\theta(z|d_i)$ become more different. Note that since we should recommend the articles with similar topic preferences to the user's preference regardless of the articles' Digg scores, we simply utilize $KL[\gamma(z|u_a)||\theta(z|d_i)]$ only for warm-start recommendations.

Top-K cold-start recommendations: We make cold-start recommendations of the three types mentioned in Section 3.2 as follows.

- For the type 1, we estimate the topic preference $\hat{\gamma}(z|u_a)$ of the active user u_a unseen in the training data and compute $Score(u_a,d_i)$ in Equation (14) by using $\theta(z|d_i)$ of the candidate article d_i seen in the training data and the estimated $\hat{\gamma}(z|u_a)$ for top-k recommendation.
- For the type 2, if an article d_i in the candidate set is unseen in the training data, we first estimate $\hat{\theta}(z|d_i)$, and compute $Score(u_a,d_i)$ using the estimated $\hat{\theta}(z|d_i)$ and the topic preference $\gamma(z|u_a)$ of the active user u_a seen in the training data.
- For the type 3, we estimate both of $\hat{\gamma}(z|u_a)$ and $\hat{\theta}(z|d_i)$ to compute $Score(u_a,d_i)$ in Equation (14).

We next present how to estimate $\gamma(z|u_a)$ for an unseen user u_a and $\theta(z|d_i)$ for an unseen Digg article d_i.

Computing $\hat{\gamma}(z|u_a)$ for an unseen user u_a: When an unseen user u_a is not included in the training data but the articles submitted or dug by u_a participate in the data, given the model parameters $\theta(z|d_i)$ of those articles d_i, the probability that u_a posts the articles in $D(u_a)$ and diggs those in $L(u_a)$ for the unknown distribution $\gamma(z|u_i)$ can be computed by using Equation (8) as

$$\prod_{d_i\in D(u_a)} e^{-\alpha KD[\gamma(z|u_a)||\theta(z|d_i)]} \cdot \prod_{d_i\in L(u_a)} e^{-(\beta/s_i)KD[\gamma(z|u_a)||\theta(z|d_i)]}, \tag{15}$$

where $\theta(z|d_i)$ is the topic preference of the article d_i which exists in the data. By Lagrangian method, we can derive the optimal distribution of $\hat{\gamma}(z|u_a)$, which maximizes the probability in Equation (15), as follows

$$\hat{\gamma}(z|u_a) = \frac{1}{Z(u_a)}\left[\prod_{d_i\in D(u_a)}\theta(z|d_i)^{\alpha} \cdot \prod_{d_i\in L(u_a)}\theta(z|d_i)^{\beta/s_i}\right]^{\frac{1}{\alpha|D(u_a)|+\sum_{d_i\in L(u_a)}\beta/s_i}}, \tag{16}$$

User	Topic z_1	Topic z_2
u_1	0.47	0.52
u_2	0.87	0.13
u_3	0.81	0.19
u_4	0.25	0.75
u_5	0.94	0.06

(a) $\gamma(z|u)$

Doc	Topic z_1	Topic z_2	Doc	Topic z_1	Topic z_2
d_1	0.11	0.89	d_5	0.43	0.57
d_2	0.87	0.13	d_6	0.70	0.30
d_3	0.96	0.04	d_7	0.31	0.69
d_4	0.93	0.06			
d_8	0.99	0.01	d_9	0.65	0.35

(b) $\theta(z|d)$

Word	Topic z_1	Topic z_2	Word	Topic z_1	Topic z_2	Word	Topic z_1	Topic z_2
Sherlock	0	0.41	Baker	0.08	0.18	Samsung	0.21	0
Holmes	0	0.20	iPhone	0.35	0	AppStore	0.14	0
221B	0	0.20	Apple	0.21	0			

(c) $\Phi(w|z)$

Doc	Seen user u_3	Seen user u_4	Unseen user u_5
d_1	0.76	-	0.56
d_2	90.7	-	-
d_3	-	-	-
d_4	-	0.66	-
d_5	3.17	15.1	1.68
d_6	-	2.31	5.91
d_7	1.86	-	1.14
d_8	2.60	0.34	15.2
d_9	-	2.98	4.38

(d) scores

Figure 6: The resulting model parameters and scores

where $Z(u_a)$ is the normalization factor to make $\sum_{k=1}^{t} \hat{\gamma}(z{=}k|u_a) = 1$. Since the articles written by u_a are not generally contained in the training data when the author u_a is not in the training data, the set $D(u_a)$ of the articles submitted by u_a is usually empty.

Computing $\hat{\theta}(z|d_i)$ for an unseen article d_i: For an unseen Digg article d_i in the training data, which was submitted by a seen user u_j and dug by seen users $U(d_i)$, we can formulate, by using Equation (8), the probability that d_i is generated to include the words in $W(d_i)$, posted by u_j and dug by the users in $U(d_i)$ as follows:

$$\alpha e^{-\alpha KD[\gamma(z|u_a)||\theta(z|d_i)]} \cdot \prod_{w\in W(d_i)} \sum_{z'\in Z} \phi(w|z')\theta(z'|d_i) \cdot \prod_{u\in U(d_i)} (\beta/s_i) e^{-(\beta/s_i)KD[\gamma(z|u)||\theta(z|d_i)]}. \quad (17)$$

To compute $\hat{\theta}(z|d_i)$ maximizing the above probability, we should develop another EM algorithm requiring heavy computation. Thus, we approximate $\hat{\theta}(z|d_i)$ by performing only the first iteration of the EM algorithm with initializing $\hat{\theta}(z|d_i)$ uniformly and calculate it as

$$\frac{1}{Z(d_i)} \left[\sum_{w\in W(d_i)} \frac{\phi(w|z)}{\sum_{z'\in Z}\phi(w|z')} + \alpha\gamma(z|u_j) + \frac{\beta}{s_i} \sum_{u\in U(d_i)} \gamma(z|u) \right], \quad (18)$$

where $Z(d_i)$ is the normalization factor to make $\sum_{k=1}^{t} \hat{\theta}(z{=}k|d_i) = 1$. Our experiments show that executing only a single iteration of the EM algorithm estimates good $\hat{\theta}(z|d_i)$s enough for cold start recommendations of both type 1 and type 3.

EXAMPLE 5.1: *Consider the Digg articles and the users shown in Figure 2. We assume that the number of topics t is 2 and $\alpha = \beta = 1$. Figure 6(a) presents the model parameters $\gamma(z|u)$, $\theta(z|d)$ and $\phi(w|z)$ after our EM algorithm finishes. Note that the hidden topics z_1 and z_2 represent the topics of 'smartphone' and 'Sherlock Holmes' respectively.*

Warm-start recommendations: *Let us find the top-1 article to recommend to the user u_3 among the candidate set C={d_1,d_2, d_5,d_7} which are not submitted nor dug by u_3. The scores of the articles in C according to Equation (14) are provided in Figure 6(d). As we discussed in Example 3.1, the user u_3 showed his interest in the topic of 'smartphone'. Since the article d_2 is on the topic 'smartphone' and obtained the highest score among the articles in C, we recommend d_2 to u_3 as the top-1 article.*

Cold-start recommendations of type 1: *The values of $\hat{\gamma}(z|u_5)$ for an unseen user u_5 according to Equation (16) are presented in Figure 6(a). Since u_5 dug the articles d_2,d_3 and d_4 on the topic 'smartphone' that is represented by z_1, $\hat{\gamma}(z_1|u_5)$ has the highest score as expected. Thus, among the candidate articles C={d_1,d_2,d_5,d_7} which are seen and not dug by u_5, d_6 has the highest score and is thus recommended as the top-1 article to u_5.*

Cold-start recommendations of type 2: *Let C={d_8,d_9} be the candidate articles which are unseen in the training data. The values of $\hat{\theta}(z|d_8)$ and $\hat{\theta}(z|d_9)$ are shown in Figure 6(b). As expected from the content of d_8, $\hat{\theta}(z_1|d_8)$ has the highest probability representing that d_8 is related the most to the topic 'smartphone'. Thus, the article d_8 obtains the highest score among the articles in C for the user u_3 and u_5. Furthermore, since d_9 is mainly dug by the users interested in the topic of 'smartphone' which is represented by z_1, we can see that $\hat{\theta}(z_1|d_9) > \hat{\theta}(z_2|d_9)$.*

Cold-start recommendations of type 3: *The recommendation scores $Score(u_a, d_i)$ between the unseen user u_5 and the unseen articles {d_8, d_9} are shown in Figure 6(d). Among the unseen articles, d_8 on the topic 'smartphone' is recommended to u_5 as the top-1 article since u_5 is interested in the same topic but d_9 is related to another topic.* ∎

Alg.	Training	Recommendation: Warm-start	Cold-start: Type 1	Cold-start: Type 2	Cold-start: Type 3
DIGTOBI	DIGTOBI-EM	DIGTOBI-W	DIGTOBI-C1	DIGTOBI-C2	DIGTOBI-C3
CTR	CTR-TR	CTR-W	CTR-C1	CTR-C2	CTR-C3
HotDigg	HOTDIGG-EM	HOTDIGG-W	-	-	-
MEM	-	MEM-W			
COS	-	COS-W	COS-C1	COS-C2	COS-C3
Baseline	-	BASE-W	BASE-C1	BASE-C2	BASE-C3

Figure 7: The implemented algorithms

6. EXPERIMENTS

We empirically evaluated the performance of our proposed algorithms. All experiments reported in this section were performed on the machine with Intel(R) Core(TM)2 Duo 2.66GHz and 2GB of main memory running Linux. The following algorithms were implemented using GCC Compiler of version 4.1.3.

- **DIGTOBI:** We implemented our proposed EM algorithm in Section 4.5 and the top-K recommendation algorithms in Section 5.
- **CTR:** Since CTR [28] combines the *matrix factorization* and the *LDA* model, and is shown to outperform the recommendations based on either matrix factorization or the LDA model only [28], we selected CTR as the state-of-the-art for this application. We downloaded the implementation of CTR in C language available at http://www.cs.princeton.edu/∼chongw/citeulike/. In [28], they proposed the recommendation algorithms for warm-start and cold-start of type 2 only. To deal with cold-start of type 1 and type 3, we estimated the latent vector u_i of the i-th user to maximize the likelihood of the *CTR* model (Equation (7) in [28]) as follows:

$$u_i = (\lambda \cdot I + \mathbb{B})^{-1} A \quad (19)$$

where $\mathbb{B}$ is a matrix such that $\mathbb{B}_{xy} = \sum_j c_{ij} v_{jx} v_{jy}$ and A is a vector such that $A_x = \sum_j c_{ij} r_{ij} v_{jx}$. Note that v_j is the latent vector of the j-th item estimated by the *CTR* model and the other parameters (i.e., c_{ij}, r_{ij} and λ_u) are the given values.

- **COS:** We implemented the recommendation algorithm in [5] as well. It was proposed to recommend Twitter messages using only the text messages and cosine similarity without any domain specific knowledge. They do not apply any user modeling method but simply represent Twitter messages of a user as a bag-of-word for content-based recommendations. Thus, this recommendation algorithm does not need the training phase.
- **HOTDIGG:** It is the implementation of HotDigg recommendation algorithm in [14]. It recommends Digg articles without considering each user's preference.
- **MEM:** We also implemented the simple memory-based recommendation algorithm in [24] which works for warm-start recommendations only.
- **BASE:** This is the implementation of the baseline algorithm which is capable of both warm and cold start recommendations by selecting K articles randomly among the candidate articles.

All algorithms for warm and cold start recommendations are categorized in Figure 7 where each empty entry denotes that its corresponding algorithm cannot handle the case represented by its column. In our experiments, the train phases of all algorithms were performs 10 times for finding local maxima with the termination condition used by *CTR* in [28] and we chose the best one as the model parameter values.

6.1 Data Sets

We downloaded 120,896 Digg articles submitted or dug by 239,847 users in two months from Dec. 2011 to Jan. 2012 using Digg API available at http://developers.digg.com/. The number of diggings, which are represented by the pairs between those downloaded users and articles, is 680,971. We removed the stop words appearing in more than 80% of all articles. Furthermore, we also deleted the words occurring in less than 3 articles since such words do not provide any clue for topical clustering. All words in Digg articles were stemmed by using the stemmer library in Lucene [16]. We refer to this data as *ORG-DATA*. We selected a subset of diggings from ORG-DATA to generate the test data sets and the rest of the other data is used as the training data to estimate the model parameters. We generated the test data sets as follows.

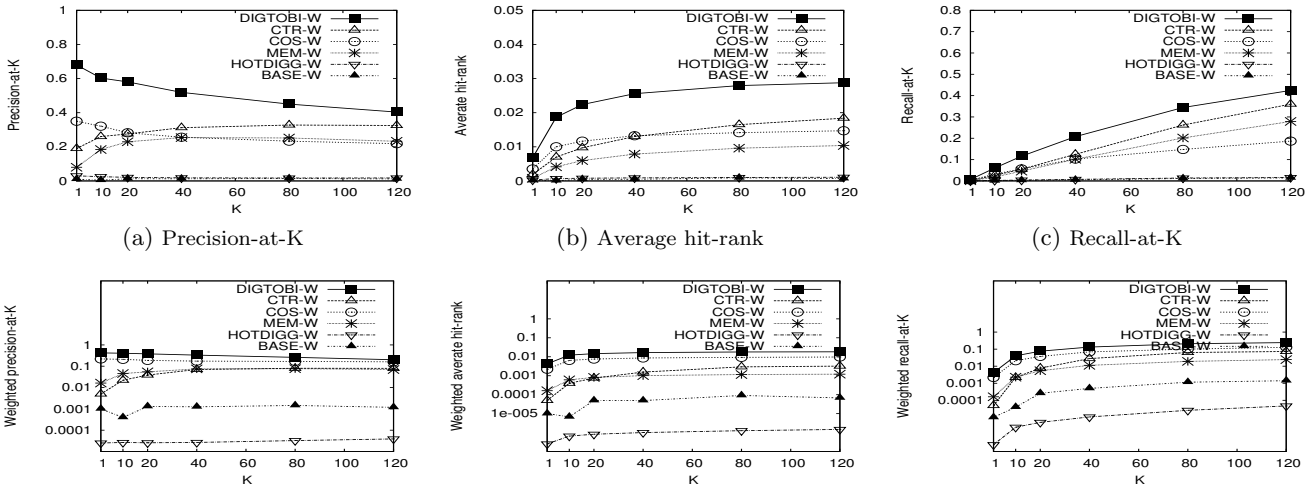
- **TEST-W1:** For warm-start recommendations, we selected 10,000 diggings of 100 users from ORG-DATA by the following steps: We first chose a seed user from ORG-DATA randomly and selected another 99 test users from ORG-DATA by choosing each user, who has the least common diggings with the users selected already, one by one greedily. Then, we extract 100 diggings of each selected user randomly. We found 7,479 number of distinct dug articles in the extracted diggings for all test users and used them as the *candidate articles* for recommendations.
- **TEST-W2:** For warm-start recommendations, we selected another set of 10,000 diggings randomly. We first choose 100 users randomly from ORG-DATA and next extracted 100 diggings for each selected user randomly. We also used distinct dug articles in the extracted diggings as the *candidate articles* for recommendations. The number of candidate articles was 4,887.
- **TEST-C1:** For cold-start recommendation of type 1, we chose 40 new users who are not included in ORG-DATA but have dug at least 10 articles in ORG-DATA. We used the articles dug by the 40 new users as the *candidate articles* for recommendations. Note that we could not select more test data for cold-start recommendations because there are a small number of new users who have dug enough articles.
- **TEST-C2:** For cold-start recommendations of type 2, we chose 40 users from ORG-DATA randomly and for each user, we downloaded 10 more Digg articles which are dug by the user but not included in ORG-DATA. Then, the newly downloaded 400 articles were used as the *candidate articles* for recommendations.
- **TEST-C3:** For cold-start recommendations of type 3, we downloaded 40 new users with 10 diggings for each user such that both of the users and their dug articles are not included in ORG-DATA. The downloaded 400 new articles are used as the *candidate articles* for recommendations.

6.2 Quality of Recommendations

We conducted our experiments with varying the number of recommendations K, the number of hidden topics t, the parameters α and β which are the constants used in the exponential KL divergence distributions in Section 4. The default values of these parameters are: K=30, t=100, α=10^6 and β=10^6.

Quality measures: We computed three basic quality measures called *recall-at-K*, *precision-at-K* [27] and *average hit-rank* [9]. (The recall-at-K is also known as *hit-rate* [9].) With the top-K recommendations for a test user u, let h be the number of correctly matched articles among the top-K recommended articles and $n_T(u)$ be the number of articles dug by the user u in the test data. Assume that d_1, ..., d_h are the h number of correctly matched articles by recommendations. Let s_i and p_i denote the Digg score of d_i and the rank of d_i among the top-K recommended articles respectively. Then, the recall-at-K and precision-at-K for u are $h/n_T(u)$ and h/K respectively. The average hit-rank is calculated as $1/n_T(u) \cdot \sum_{i=1}^{h} 1/p_i$ which measures the effectiveness of ranking for each test user. As the dug articles by a test user appear with higher ranks in the top-K recommended articles, this measure becomes larger. Thus, the high values of average hit-rank are more desirable.

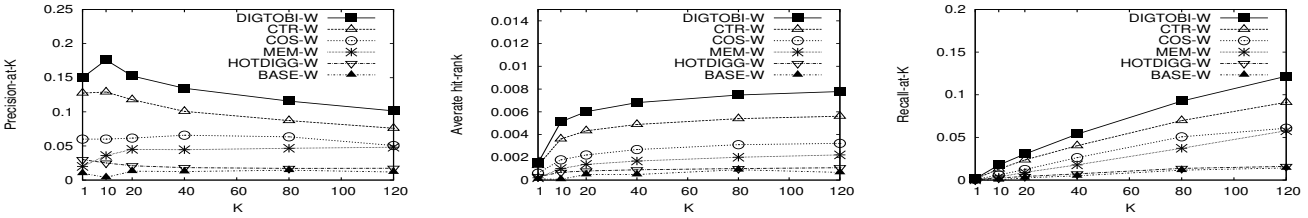
Digg articles with high scores tend to be exposed in the front pages of Digg service and thus get more chances of being dug. However, it is also desirable to recommend the relevant articles with low Digg scores to user's preference. Thus, we revised the above three goodness measures and also calculated those new measures to show that DIGTOBI does not simply select popular articles but finds each user's favorite articles among the low scored articles as well. To have better scores for the correctly recommended articles with small Digg scores, when s_i denotes the Digg score of the correctly matched article d_i, the *weighted* recall-at-K is calculated as $1/n_T(u) \cdot \sum_{i=1}^{h} 1/s_i$ which is the sum of the inverse of the Digg score for every correctly recommended article. Similarly, the *weighted* precision-at-K and *weighted* average hit-rank are calculated as $1/K \cdot \sum_{i=1}^{h} 1/s_i$ and $1/n_T(u) \cdot \sum_{i=1}^{h} 1/(p_i \cdot s_i)$ respectively.



(a) Precision-at-K (b) Average hit-rank (c) Recall-at-K

(d) Weighted precision-at-K (e) Weighted average hit-rank (f) Weighted recall-at-K

Figure 8: Top-K Digg article recommendations with varying K (*TEST-W1*)



(a) Precision-at-K (b) Average hit-rank (c) Recall-at-K

Figure 9: Top-K Digg article recommendations with varying K (*TEST-W2*)

Warm-start recommendations: We first evaluate the quality of warm-start recommendations using *TEST-W1*. Figures 8(a)–(c) show *precision-at-K*, *average hit-rank* and *recall-at-K* of the six warm-start recommendation algorithms listed in Figure 7 respectively with varying K from 1 to 120.

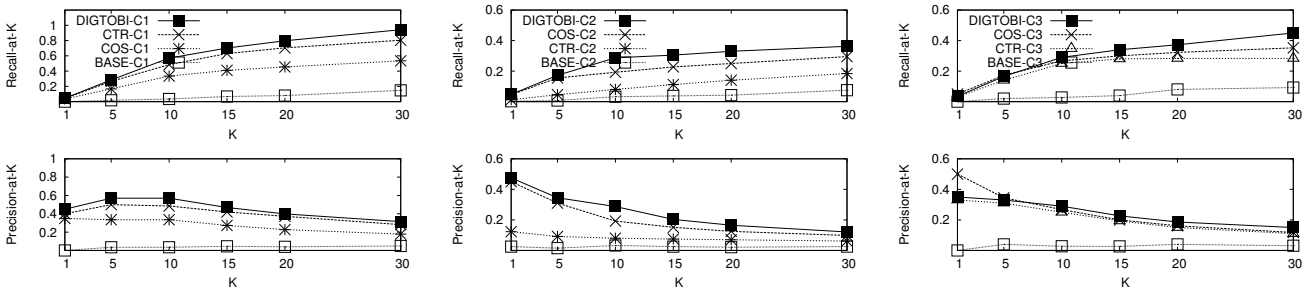
The graphs illustrate that *DIGTOBI-W* for every K outperforms the other recommendation algorithms in terms of every quality measure. As expected, *HOTDIGG* and *BASE* are the worst performers since both of them are not personalized recommendation algorithms.

As we increase K, since we have more chances to answer the right articles correctly, both recall-at-K and average hit-rank of all algorithms grow gradually. The precision-at-K of *DIGTOBI-W* is the largest when K=1 and decreases gradually as K is increased. However, the precision-at-K of the other algorithms increases with growing K. This is because *DIGTOBI-W* has good characteristics of making the correct answers to have higher ranks while the other algorithms do not. For the same reason, the performance improvement of our algorithm *DIGTOBI-W* to the other algorithms becomes better with respect to the average hit-rank rather than the recall-at-K for a fixed K. For instance, when K=10, *DIGTOBI-W* shows better performance than *CTR-W* by 2.32 times with the recall-at-K, but with the average hit-rank, *DIGTOBI-W* outperforms *CTR-W* by 3.07 times.

With the same experiments, we also plotted the three new weighed quality measures in Figure 8(d)–(f). The log scale was used in the y-axis. The graphs show that the recommendations by *DIGTOBI-W* show better qualities than those by the other algorithms for every K. *COS-W* and *CTR-W* are the second and the third best performers respectively. For example, when K=10, the modified precision-at-K of *DIGTOBI-W* is 1.92 times higher than that of *COS-W* and the modified average hit-rank of *DIGTOBI-W* is 1.89 times better than that of *COS-W*. Furthermore, when K=10, the modified precision-at-K of *DIGTOBI-W* is 23.9 times higher than that of *CTR-W* and the modified average hit-rank of *DIGTOBI-W* is 40.5 times better than that of *CTR-W*. Thus, we conclude that *DIGTOBI-W* recommends the relevant articles to each user's preference effectively even though they have low Digg scores.

We also tested with *TEST-W2* and plotted the precision-at-K, average hit-rank and recall-at-K in Figure 9(a)–(c). The quality of recommendations shows similar trends with the results of using *TEST-W1*. Since the graphs for the weighted quality measures also show similar trends, we do not present the graphs with *TEST-W2* here. In general, the values of all quality measures with *TEST-W2* are smaller than those with *TEST-W1*. The reason is as follows: For *TEST-W1*, since we selected the users having a small number of common diggings as possible, it is more likely that the candidate articles not dug by a test user u_a is the one in which u_a is actually not interested. However, for *TEST-W2*, since we selected the test data randomly, the candidate articles probably include the articles which are not dug by a test user u_a but relevant to the preference of u_a. Even though such articles are recommended to the user u_a, we have to regard them as incorrect answers. Thus, the values of all quality measures with *TEST-W2* become smaller than those with *TEST-W1*.

Cold-start recommendations: Using *TEST-C1*, we show the recall-at-K and precision-at-K of *DIGTOBI-C1*, *CTR-*



(a) Type 1 (b) Type 2 (c) Type 3

Figure 10: Top-K cold-start recommendations with varying K

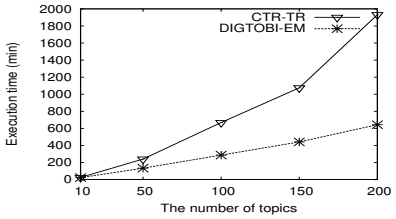


Figure 11: Execution time varying t

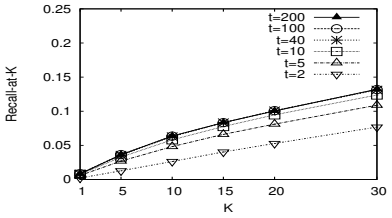


Figure 12: Recall-at-K varying t

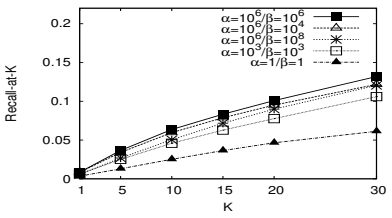


Figure 13: Recall-at-K varying (α,β)

C1, *COS-C1* and *BASE-C1* with increasing K from 1 to 30 in Figure 10(a). Note that *MEM* cannot handle any type of cold-start recommendation. We can see that *DIGTOBI-C1* makes the best recommendations to the test users in *TEST-C1*, even though the test users are not considered in the phase of learning model parameters. For every value of K, *CTR-C1* shows the second best performance in terms of both recall-at-K and precision-at-K.

With *TEST-C2*, we evaluate the qualities of cold-start recommendations of type 2 by *DIGTOBI-C2*, *CTR-C2*, *COS-C1* and *BASE-C2*, and plot both quality measures in Figure 10(b). Remember that the type 2 recommends the Digg articles, which are not seen in the training data, to the seen users included in the training data. The graph shows that *DIGTOBI-C2* also outperforms the other algorithms for cold-start recommendations of type 2.

Figure 10(c) shows the recall-at-K and precision-at-K of *DIGTOBI-C3*, *CTR-C3*, *COS-C3* and *BASE-C3* using *TEST-C3* for cold-start recommendation of type 3. Here, we recommend Digg articles to the test users in *TEST-C3* where both articles and test users are not included in the training data. The graph shows that *COS-C3* is better than *DIGTOBI-C3* with $K \leq 5$ but our algorithm *DIGTOBI-C3* shows better performance with the other values of K.

Execution time for estimating model parameter: In Figure 11, with varying the number of topics t from 2 to 200, we plotted the running times of the inference algorithms *DIGTOBI-EM* and *CTR-TR*. The graph shows that as the number of topics t is increased, the execution time of our inference algorithm *DIGTOBI-EM* grows linearly with t. However, the speed of *CTR-TR* slows down very fast as t becomes larger. We conclude that *DIGTOBI-EM* outperforms the variational EM algorithm of the LDA model used in *CTR-TR* in terms of speed for estimating model parameters.

Varying t: With varying K from 1 to 30 and the number of topics t from 2 to 200 together, we measured the quality of recommendations by our recommendation algorithm *DIGTOBI-W* in terms of the recall-at-K with the test data *TEST-W1* as shown in Figure 12. The graph shows that *DIGTOBI-W* obtains the best quality of recommendations when $t \geq 100$ and the quality of recommendations does not improve any more with $t > 100$. Thus, we use $t=100$ for the default value in our experiments.

Varying α and β: To find the best values for the constants α and β in our probabilistic model presented in Section 4.2, we varied α and β together and plotted the recall-at-K with our recommendation algorithm *DIGTOBI-W* in Figure 13. The quality of recommendations by our algorithm was the best when $\alpha=10^6$ and $\beta=10^6$. Thus, we set the default values of both α and β to 10^6 in our experiments.

7. CONCLUSION

We presented DIGTOBI, a personalized recommendation system for Digg articles using a novel probabilistic modeling. Utilizing the observations that Digg users submit their Digg articles and vote thumbs up for the Digg articles submitted by others depending on their topic preferences, we designed our generative model to describe the probabilistic processes of submitting and digging articles by each user. Our model improves the quality of recommendations by enabling the relevant articles with low Digg scores as well as high Digg scores to be considered important. We developed the EM algorithm for learning the best model parameters in our probabilistic model. We also provided effective warm-start and cold-start recommendation algorithms utilizing our model parameters. By performance study, we confirmed the effectiveness of DIGTOBI by comparing the performance with the traditional recommendation algorithms.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No. 2012-0000111). This was also supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2012-033342).

8. REFERENCES

[1] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD*, 2009.

[2] D. Agarwal and B.-C. Chen. fLDA: matrix factorization through latent dirichlet allocation. In *WSDM*, pages 91–100, 2010.

[3] D. P. Bertsekas. *Nonlinear Programming (Second ed.)*. Cambridge, 1999.

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

[5] J. Chen, R. Nairn, L. Nelson, M. S. Bernstein, and E. H. Chi. Short and tweet: experiments on recommending content from information streams. In *CHI*, 2010.

[6] CiteULike. http://www.citeulike.org.

[7] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, 1991.

[8] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statist. Soc.*, 39:1–38, 1977.

[9] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[10] Digg. http://digg.com, 2011.

[11] Facebook. http://www.facebook.com.

[12] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, 1999.

[13] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1), 2004.

[14] Y. Kim and K. Shim. HotDigg: Finding recent hot topics from digg. In *DASFAA*, 2012.

[15] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML*, pages 577–584, 2006.

[16] Lucene. http://lucene.apache.org.

[17] B. Mehta, T. Hofmann, and W. Nejdl. Robust collaborative filtering. In *RecSys*, 2007.

[18] Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. Topic sentiment mixture: modeling facets and opinions in weblogs. In *WWW*, 2007.

[19] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[20] Netflix. http://www.netflix.com.

[21] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys*, 2009.

[22] M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[23] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[24] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[25] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

[26] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML*, pages 704–711, 2003.

[27] B.-Q. Vuong, E.-P. Lim, A. Sun, M.-T. Le, and H. W. Lauw. On ranking controversies in wikipedia: models and evaluation. In *WSDM*, 2008.

[28] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, 2011.

[29] Wikipedia. Geometric mean. http://en.wikipedia.org/wiki/Geometric_mean.

[30] Wikipedia. Bernoulli distribution. http://en.wikipedia.org/wiki/Bernoulli_distribution.

[31] C. F. J. Wu. On the convergence properties of the em algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.