Incremental Probabilistic Latent Semantic Analysis for Automatic Question Recommendation

Hu Wu
Institute of Software, Chinese Academy of Sciences
wuhu@itechs.iscas.ac.cn

Yongji Wang
Institute of Software, Chinese Academy of Sciences
ywang@itechs.iscas.ac.cn

Xiang Cheng
Peking University
cxwcfea81@yahoo.com.cn

ABSTRACT

With the fast development of web 2.0, user-centric publishing and knowledge management platforms, such as Wiki, Blogs, and Q & A systems attract a large number of users. Given the availability of the huge amount of meaningful user generated content, incremental model based recommendation techniques can be employed to improve users' experience using automatic recommendations. In this paper, we propose an incremental recommendation algorithm based on Probabilistic Latent Semantic Analysis (PLSA). The proposed algorithm can consider not only the users' long-term and short-term interests, but also users' negative and positive feedback. We compare the proposed method with several baseline methods using a real-world Question & Answer website called Wenda. Experiments demonstrate both the effectiveness and the efficiency of the proposed methods.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Information Search and Retrieval-Information filtering

General Terms

Algorithms, Experimentation

Keywords

Incremental learning, PLSA, Recommendation System

1. INTRODUCTION

Social-network products are flourishing. Sites such as MySpace, Facebook, Orkut, and Yahoo! Answers attract millions of users a day. The rapid growth of the amount of users and items on social-network sites has made information finding increasingly challenging. Content-based recommendation tries to solve the challenge by recommending items similar to those that a given user has liked in the past, whereas in Collaborative Filtering (CF) one identifies user whose tastes are similar to those of the given and recommended items they have liked.

RecSys'08, October 23–25, 2008, Lausanne,Switzerland.

For a typical recommender system, it is common that the old data (both users and items) keep changing, and the new data become available continually, such as articles in Google news [12]. In this case, we might retrain the model using both old and new data. However, it is infeasible in practice since batch training is computationally expensive. To speed up the performance, we need incremental Collaborative Filtering (ICF) [4] methods that can efficiently handle new data arriving in a stream, instead of retraining the whole model from the scratches. The incremental ability of a recommendation algorithm can not only reduce its computational cost, but also makes it applicable to large-scale data sets.

In this work, we focus on an incremental learning recommendation method using Probabilistic Latent Semantic Analysis (PLSA) for automatic question recommendation in a Question & Answer (Q & A) website. Probabilistic Latent Semantic Analysis (PLSA) is widely used in both content based recommendation [1] and collaborative filtering [6]. It is based on the observation that user preference and item characteristics are often governed by a few latent semantics. To be more specific, PLSA introduces a latent variable, and decouples the probabilistic dependency between users and items into the dependency between users and latent semantics and the dependency between the latent semantics and items, both in a probabilistic way. Moreover, the probabilistic model could be learned using the Expectation Maximization (EM) algorithm [3], and the convergence is guaranteed [16].

Although PLSA has been successfully developed, there are two main shortcomings. First, the PLSA model is estimated only for those documents appearing in the training set. Each document is seen as a random mixture over latent topics. Latent Dirichlet Allocation (LDA) was proposed to deal with the weakness of PLSA, where LDA parameters were estimated by the approximate inference algorithms, such as variational EM and Gibbs Sampling. PLSA was shown to be a special variant of LDA with a uniform Dirichlet prior in a maximum a posteriori model [2]. Secondly, PLSA lacks the incremental ability, *i.e.* it cannot handle new data arriving in a stream. To handle streaming data, a naive approach is that we can re-train the model using both existing training data and new data. However, it is apparently not efficiently since it is very computationally expensive. What is more, for some practical applications, this is infeasible since the system needs real-time online update. Therefore, we need a fast incremental algorithm without compromising recommendation quality.

In this paper, we propose an incremental PLSA algorithm. The advantages of the proposed method can be summarized as:

- Our method is very fast, so that it can be deployed in real-time recommender systems. To update the model for a new question, our algorithm needs 0.54 sec. on average, which can meet the real-time requirement.

- The proposed algorithm can take into account both the users' long-term and short-term interests. The long-term interests are reflected from all the questions that users already asked, while short-term interested are reflected from the new questions he just asked lately.
- The proposed algorithm enjoys the flexibility to make updates based on users' positive and negative feedback. If a user gives a high rate to a recommended item, we regard it as a positive feedback, otherwise it is a negative feedback.

We study the proposed method using the data set from a real-world Question & Answer website, compared with serval baseline methods. Experiments demonstrate both the effectiveness and efficiency of the proposed methods.

The remainder of this paper is organized as follows: In Section 2 we describe Question & Answer systems, and the problem of automatic question recommendation in Q & A systems. In Section 3 we review basic ideas of PLSA, and describe how to recommend relevant questions using PLSA. In Section 4, we review previous work on incremental PLSA algorithms and point out the problems with these existing methods. In Section 5 we present the proposed incremental methods. In Section 6 we compare the proposed method against four baseline methods using a Q & A data set. We then conclude the paper and point out future research directions in Section 7.

2. QUESTION AND ANSWER (Q & A) SYSTEM

Question & Answer websites (referred to as Q & A system afterwards) are becoming popular in recent years, such as Yahoo! Answers [1], Baidu Zhidao [2] *etc.*. In Q & A systems, users post questions to seek help from others. In the meanwhile, users also answer others' questions. These sites are becoming huge knowledge bases in Internet, thus attract millions of users who either ask questions to seek help or reply questions to seek pleasure by helping others [19].

Wenda [3] is a Chinese Q & A website launched by Google recently. Currently, Wenda has a hundred thousand users, and the number of users is still increasing. Given the huge number of users and questions, there is a challenge problem: it is hard for a user to precisely and quickly locate the questions and answers that might interest him [17]. To solve this problem, we implement automatic question recommend algorithms in Wenda, *i.e.*, when a user views a question, the system automatically displays related questions to the user based on the questions (answers) he posted and the question he is viewing. The recommendation mechanism is illustrated in Figure 1 (Left). There are two scenarios that incremental learning should be considered in Q & A systems: (i) A new user registers in the system; and (ii) A new question (answer) is posted by existing users.

For a good recommendation system, we also need to consider two facts. First, users have both long-term and short-term interests. We can learn users' long-term interests by accumulating users' preference for a long period. So users' long-term interests are relatively stable. On the other hand, users' short-term interests play a more important role on *instant* recommendation, although they are instantaneous and unstable. Our algorithm needs to capture both long-term and short-term interests of users. Secondly, users can give both positive and negative feedback to the recommended items. If the user gives a high score on a given recommended item, we refer this as *positive feedback*, otherwise *negative feedback*. Our algorithm needs to learn both positive and negative feedback from users. The model updating flow is illustrated in Figure 1 (Right).

We next introduce automatic question recommendation using Probabilistic Latent Semantic Analysis (PLSA).

[1] http://answers.yahoo.com
[2] http://zhidao.baidu.com
[3] http://wenda.tianya.cn

3. AUTOMATIC QUESTION RECOMMENDATION USING PLSA

In this section, we describe automatic question recommendation using Probabilistic Latent Semantic Analysis (PLSA). We first give a brief introduction to PLSA algorithm, and then describe how to apply it to automatic question recommendation.

3.1 Probabilistic Latent Semantic Analysis

For question recommendation tasks, it is reasonable to assume the following approximation: the word is independent of the user when a user wants to express some certain meaning (i.e. the latent semantics is known), so when the latent semantics under the questions and answers are found, we are able to make recommendation based on similarities on these latent semantics. Therefore, PLSA could be used to model the users' profile (represented by the questions that the user asks or answers) and the questions as well through estimating the probabilities of the latent topics behind the words. Because the user's profile is represented by all the questions that he/she asks or answers, we only need to consider how to model the question properly.

Suppose we have N questions denoted as Q and M words in the dictionary denoted as W. In addition, in order to capture the latent semantics, K latent topics are introduced notated by $< z_1, z_2, ..., z_K >$. Then we consider a user as a document that contains the words that he used in his questions or answers. A user's interest is thus represented by many dyadic pairs $< qustion_i, word_j >$ where i and j are their index in the user set and word dictionary, respectively. Based on the independent assumption made above, we could rewrite the probability of co-occurrence $< q_i, w_j >$ as follows:

$$P(q_i, w_j) = P(q_i) \sum_{z_k} P(z_k|q_i) P(w_j|z_k). \quad (1)$$

Supposing a uniform distribution of $P(q_i)$ for all the questions, the learning task is boiled down to learning $P(z_k|q_i)$ and $P(w_j|z_k)$ and to update them accordingly when the user's profile changes. [1] provided a Expectation Maximization (EM) method for the PLSA model fitting.

- E-Step: compute posterior probabilities for the latent variables

$$P(z_k|q_i, w_j) = \frac{P(w_j|z_k)P(z_k|q_i)}{\sum_{l=1}^{K} P(w_j|z_l)P(z_l|q_i)}, \quad (2)$$

- M-Step: maximize the expected complete data log-likelihood

$$P(w_j|z_k) = \frac{\sum_{i=1}^{N} n(q_i, w_j)P(z_k|q_i, w_j)}{\sum_{m=1}^{M}\sum_{i=1}^{N} n(q_i, w_m)P(z_k|q_i, w_m)}, \quad (3)$$

$$P(z_k|q_i) = \frac{\sum_{j=1}^{M} n(q_i, w_j)P(z_k|q_i, w_j)}{n(q_i)}. \quad (4)$$

The learning process is iterating the E-Step and M-Step alternatively until some convergence condition (such as Log likelihood) is satisfied. Typically, 20-50 iterations are needed before converging [1]. The problem is that, with this algorithm, whenever user's profile changes, the whole model needs to be retrained from scratch.

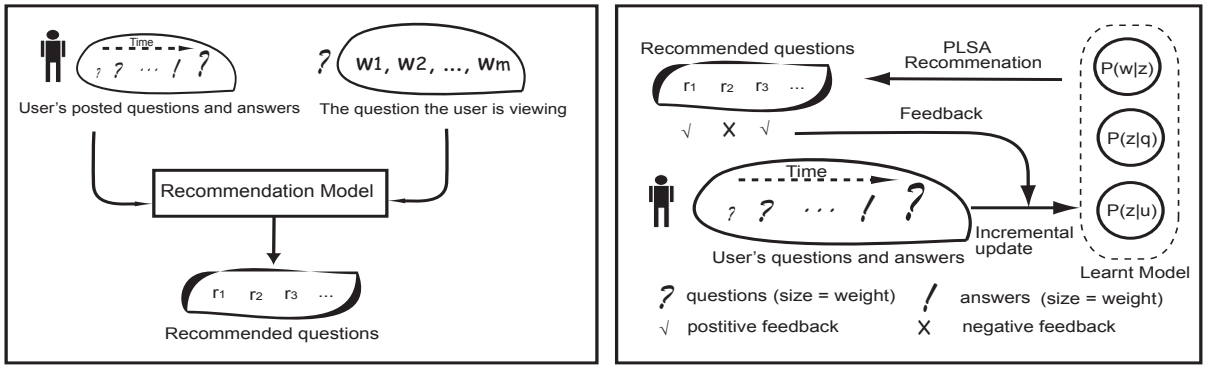

Figure 1: Question recommendation in the Wenda system. Left: Recommendation based on the user's history posts and the question that he is currently viewing. Right: Incremental learning model update considering the user's long-term and short-term interests and feedback about our recommendations.

This makes the update process could only be done offline. In the next section, we will see some existing work addressing this problem.

3.2 Automatic Question Recommendation

To generate the recommended question list, we first learn $P(z|q)$, the probability of the latent topic given a question, using all the questions that user has asked or answered. We then calculate $P(z|u)$, the probability of the latent topic given a user, using the following Equation 5:

$$P(z|u) = \frac{1}{L}\sum_{i=1}^{L} P(z|q_i), \tag{5}$$

where $\{q_1, q_2, ..., q_L\}$ is the list of questions that the user has asked or answered.

Based on $P(z|u)$ and $P(z|q)$, we can calculate two similarity measures, namely *question-question similarity* and *user-question similarity* using the inner product as follows:

- Similarity between the question that the user is currently viewing q_c and all the other questions q_j:

$$S_{q_c,q_j} = \sum_z P(z|q_c)P(z|q_j), \tag{6}$$

- Similarity between user u_i and question q_j:

$$S_{u_i,q_j} = \sum_z P(z|u_i)P(z|q_j). \tag{7}$$

Given the similarity between user u_i and question q_j: S_{u_i,q_j}, and the similarity between other questions and the question that the user is viewing: S_{q_c,q_j}, we could recommend the most related questions by simply sorting the scores computed as:

$$Score_{q_j} = S_{u_i,q_j} + S_{q_c,q_j}. \tag{8}$$

Note that the user-question and question-question similarities are computed online. Therefore, the recommendation list changes instantly whenever a user is viewing a different question or the user posts a new question or answer.

However, PLSA model retraining is a computational intensive task that can not be done in real time. We now proceed to explain how to retrain the model more quickly with the help of incremental learning.

4. EXISTING PLSA INCREMENTAL METHODS

Incremental ability is essential when the training examples in practical recommendation systems become available over time, usually one at a time [4].

There are some existing work on incremental learning of PLSA. [1] provided a simple update scheme called **Fold-In**. The main idea is to update the $P(z|q)$ part of the model while keeping $P(w|z)$ fixed. However, $P(w|z)$ can change significantly during EM iteration and affect $P(z|q)$ as well. Thus, the result of Fold-In might be biased.

Tzu-Chuan Chou *et al* [5] proposed **Incremental PLSA** (**IPLSA**), a complete Bayesian solution aiming to address the problem of online event detection. For the automatic question recommendation task, when a new question is posted, both $P(z|q)$ and $P(w|z)$ are updated as follows:

1. Fold in new questions:

$$P(z|w, q_{new}) = \frac{P(w|z)P(z|q_{new})}{\sum_{z'\in Z} P(w|z')P(z'|q_{new})}, \tag{9}$$

$$P(z|q_{new}) = \frac{\sum_{w\in q_{new}} n(w,q_{new})P(z|w,q_{new})}{\sum_{z'\in Z}\sum_{w\in q_{new}} n(w,q_{new})P(z'|w,q_{new})}. \tag{10}$$

2. Fold in new words:

$$P(z|w_{new}, q_{new}) = \frac{P(q_{new}|z)P(z|w_{new})}{\sum_{z'\in Z} P(q_{new}|z')P(z'|w_{new})}, \tag{11}$$

$$P(z|w_{new}) = \frac{\sum_{q\in Q_{new}} n(w_{new},q)P(z|w_{new},q)}{\sum_{q'\in Q_{new}} n(w_{new},q')}. \tag{12}$$

3. Update PLSA parameters, all $P(w|z)$ are normalized using the following formula:

$$P(w|z) = \frac{\sum_{q\in Q\cup Q_{new}} n(w,d)P(z|w,q)}{\sum_{q'\in Q\cup Q_{new}}\sum_{w'\in q'} n(w',q')P(z|w',q')}. \tag{13}$$

For the time complexity, the algorithm needs $O(n_{iter}\cdot(n_{nq}+n_{oq})\cdot(n_{nw}+n_{ow})\cdot K)$ operations to converge whenever there are new questions added, where n_{nq} is the number of new questions, and n_{oq} is the number of old questions, and n_{nw} is the number of new words and n_{ow} is the number of old words, and K is the number of latent topics, and n_{iter} is the number of iterations. Note that the computational complexity is the same as that of the batched PLSA algorithm, although less EM iterations are needed.

Table 1: Detailed qualitative comparison results of the four models.

	Fold-in	IPLSA	MAP-PLSA	Our method
Updated parts during incremental learning	$P(z\|u)$	$P(z\|u)$ and $P(w\|z)$	$P(z\|u)$ and $P(w\|z)$	$P(z\|u)$ and $P(w\|z)$
Ability of handling new words during update	No	Yes	Yes	Yes
Accuracy	Low	Low	Medium	High
Incremental complexity	Low	High	High	Medium
Flexibility	Low	Low	Low	High
Adaptable to user's feedback	No	No	No	Yes

Chien and Wu proposed another PLSA incremental learning algorithm named **MAP-PLSA** [10]. Different from the traditional PLSA learning formula derived using Maximum Likelihood (ML) assumption, MAP-PLSA updates PLSA parameters using the Maximum A Posterior (MAP) as follows:

$$\hat{P}_{MAP}(w_j|z_k) = \frac{\sum_{i=1}^{N} n(q_i, w_j)P(z_k|q_i, w_j) + (\alpha_{j,k}^{(n-1)} - 1)}{\sum_{m=1}^{M}[\sum_{i=1}^{N} n(q_i, w_m)P(z_k|q_i, w_m) + (\alpha_{m,k}^{(n-1)} - 1)]}, \tag{14}$$

$$\hat{P}_{MAP}(z_k|q_i) = \frac{\sum_{j=1}^{M} n(q_i, w_j)P(z_k|q_i, w_j) + (\beta_{k,i}^{(n-1)} - 1)}{n(q_i) + \sum_{l=1}^{K}(\beta_{l,i}^{(n-1)} - 1)}, \tag{15}$$

where

$$\alpha_{j,k}^{(n)} = \sum_{i=1}^{N_n} n(q_i^{(n)}, w_j^{(n)})P^{(n)}(z_k|q_i^{(n)}, w_j^{(n)}) + \alpha_{j,k}^{(n-1)}, \tag{16}$$

$$\alpha_{j,k}^{(0)} = 1 + \sum_{i=1}^{N} n(q_i, w_j)P(z_k|q_i, w_j), \tag{17}$$

$$\beta_{k,i}^{(n)} = \sum_{j=1}^{M_n} n(q_i^{(n)}, w_j^{(n)})P^{(n)}(z_k|q_i^{(n)}, w_j^{(n)}) + \beta_{k,i}^{(n-1)}, \tag{18}$$

$$\beta_{k,i}^{(0)} = 1 + \sum_{j=1}^{M} n(q_i, w_j)P(z_k|q_i, w_j). \tag{19}$$

The advantage of MAP-PLSA is its update efficiency. The time complexity is $O(n_{iter} \cdot n_{nq} \cdot \|nq\| \cdot K)$, where $\|nq\|$ is the average number of words of new questions. But the results can also be biased, especially for $P(w|z)$ according to Equation (14).

Besides the above work, [12], [9], and [7] modified the original PLSA model and provided some experimental results on how to achieve the balance between efficiency and accuracy. Banerjee and Basu proposed online variants of other probabilistic model such as LDA [15] for news clustering. [8] proposed a novel incremental algorithm based on non-parametrical Dirichlet Process for the new topic detection problem.

Table 1 summarizes detailed qualitative comparisons among these algorithms and our proposed incremental PLSA algorithm. The "Accuracy" measure is the degree of approximation to the batched PLSA; this measure affects recommendation precision in our experiments. "Incremental Complexity" is the measure of efficiency of the update algorithm; "Flexibility" is the measure of whether the model can reflect the user's latest interest while still reflecting user's long-term interest. "Adaptable to User's Feedback" indicates whether the model can update according to the user's feedback.

We next describe our proposed incremental PLSA methods.

5. OUR METHOD

There are three issues we need to consider for the incremental task in Q & A systems:

- We need to adjust both word-topic and user-topic probabilities for new posted questions (answers).
- We need to consider both users' short-term and long-term interests while updating the model parameters.
- We need to consider both users' positive and negative feedback.

We next propose a novel incremental PLSA learning algorithm that address the above three problems.

When a user u posts a new question or answers an existing question q, the probability of a latent topic given the user $P(z|u)$ is updated accordingly, and so does the probability of words given a topic, $P(w|z)$. We propose a modified EM scheme based on the Generalized Expectation Maximization (GEM) [14]. The formulae for incremental update are as follows:

- E-Step:

$$P(z|q, w)^{(n)} = \frac{P(z|q)^{(n)}P(w|z)^{(n)}}{\sum_{z'} P(z'|q)^{(n)}P(w|z')^{(n)}}. \tag{20}$$

- M-Step:

$$P(z|q)^{(n)} = \frac{\sum_{w} n(q, w) \times P(z|q, w)^{(n)}}{\sum_{z'}\sum_{w'} n(q, w') \times P(z'|q, w')^{(n)}}, \tag{21}$$

$$P(w|z)^{(n)} = \frac{\sum_{q} n(q, w) \times P(z|q, w)^{(n)} + \alpha \times P(w|z)^{(n-1)}}{\sum_{q}\sum_{w'} n(q, w') \times P(z|q, w')^{(n)} + \alpha \times \sum_{\tilde{w}} P(\tilde{w}|z)^{(n-1)}}, \tag{22}$$

where the superscript $(n-1)$ denotes the old model parameters and (n) for the new ones, $w' \in q_w$ and $\tilde{w} \in W$ are words in this question and all other words in the dictionary, respectively.

After several EM iterations, we can get a stable value of $P(z|q)$. After that, $P(z|u)$ could be calculated as:

$$P(z|u)^{(n)} \propto P(z|u)^{(n-1)} + \beta \times P(z|q), \tag{23}$$

where $P(z|u)^{(0)}$ and $P(w|z)^{(0)}$ are initialized randomly for all users and words. The values of α and β are hyper-parameters that manually selected based on empirical results (see Section **??**). The detailed algorithm description are shown in Algorithm 1.

The time complexity of Algorithm 1 is $O(n_{iter} \cdot n_{nq} \cdot \|n_{nq}\| \cdot K + \|u_q\| \cdot K)$, where n_{iter} is the number of iterations, n_{nq} is the number of new questions, $\|n_{nq}\|$ is the average number of words in these questions and $\|u_q\|$ is how many users are involved in the discussion of this question plus the number of users who provide feedback about this question.

The major advantage of the proposed algorithm is that we can take into account two different scenarios by adjusting the value of weight β.

In the first scenario, we want to consider both users' long-term and short-term Interests. Suppose user u has posted N_u questions (answers) before, and q is a new question posted by the user. Intuitively, we can set the weight on question q as $\frac{1}{N_u+1}$ in order to

Algorithm 1: Our Incremental PLSA learning algorithm.

Input: New question q, $P(z|u)$ of the author u for all the latent topics z, $P(w|z)$ for all the words w in the new question
Output: For all the topics, output updated $P(z|u)$ for the author, updated $P(w|z)$ for all the words, new probabilities $P(z|q)$ of the new question

```
if user u is new then
    for all the z do
        Randomize and normalize P(z|u)^(n-1) to ensure that ∑_z P(z|u)^(n-1)=1 ;
    end
end
else
    for all the z do
        P(z|q) = P(z|u)^(n-1);
    end
end
for All the words w appear in the new question do
    if word w is new then
        for all the z do
            Randomize P(w|z) and ensure ∑_w P(w|z) = 1;
        end
    end
end
while not convergent do
    for all the latent topics z do
        for all the < q, w > pairs for all the words in the question q do
            P(z = k|q, w) = P(z=k|q)P(w|z=k) / ∑_z' P(z'|q)P(w|z') ; // E-Step
        end
    end
    for all the latent topics z do
        P(z = k|q) = ∑_w n(q,w)×P(z=k|q,w) / ∑_w,z' n(q,w)×P(z'|q,w) ; // M-Step
    end
    for all the latent topics z do
        for all the words w in the question q do
            P(w|z) = n(q,w)×P(z|q,w)+α×P(w|z) / ∑_w' n(q,w')×P(z|q,w')+α×∑_ŵ P(w|z)^(n-1) ; //M-Step
        end
    end
end
for all the authors u in the question q do
    for all the topics z do
        P(z|u)^(n) = P(z|u)^(n-1) + β × P(z|q);
        Normalize P(z|u)^(n);
    end
end
```

achieve the balance between existing and new questions. However, to better reflect users's short-term interests, we might increase the weight β on question q. Therefore, β needs to be larger than $\frac{1}{N_u+1}$. Therefore, we can set a higher β value for new questions to better reflect users' short-term interests.

In the second scenario, in Q & A systems, the user can give positive or negative rating to a recommended question, which indicates whether he is interested in the given question or not. Users' feedback is important and we need consider this information in our incremental algorithm. For the positive feedback, we set the weight β a positive value. Otherwise, we set the weight β a negative value. Note that the probability value might be negative if we set β negative. We need a shift operation to make the value of all probabilities larger than zero, and then a normalization operation to ensure the sum of all probabilities equals to one. Therefore, we can take both users' positive and negative feedback into considerations.

We next apply the proposed incremental algorithm to a real-world data set from the Wenda website.

6. EXPERIMENTS

We implemented our proposed method together with other incremental PLSA implementations for the question recommendation task in the Wenda system. In this section, we give the performance details of these algorithms.

6.1 The Data Set

Our data set contains the questions and corresponding answers of users who registered in the Wenda website from October 2007 to April 2008. The details of the data set are shown in Table 2.

Table 2: The Statistical Information of the Evaluation Data Set.

Item	Number
Users	108300
Average posts per user	1.64
Questions	38375
Answers	138906
Words †	62187
Average question length ‡	189 (words)
Feedback	7122

† After word segmentation ‡ Including all the answers to the questions

We apply two preprocessing steps on all the questions and answers: Chinese word segmentation and stop-word filtering. Finally, we get 62187 words and the average question length is 189 words. There are 7122 feedback provided by Wenda users about the recommended questions. The average number of answers per question is around 3.62. Figure 2 shows the total number of questions and answers in the Wenda website since its launch.

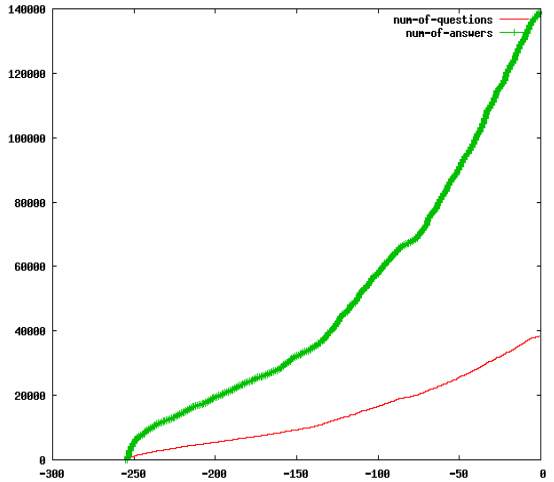

Figure 2: The increase of the number of questions and answers in Wenda. The x-coordinate is the number of days before 2008.4.30 and the y-coordinate is the number of questions and answers.

Table 3: Update time (in seconds) comparison among four incremental schemes and batched PLSA on Wenda data set.

Iterations	Batched PLSA		Fold-In		IPLSA		MAP-PLSA		Our method	
	Total Time	One Question	Total Time	One Question	Total Time	One Question	Total Time	One Question	Total Time	One Question
5	266	0.54	**34**	**0.06**	99	0.19	88	0.17	80	0.16
10	528	1.04	**69**	**0.13**	195	0.38	157	0.31	148	0.28
15	796	1.58	**103**	**0.20**	292	0.58	223	0.44	213	0.42
20	1062	2.12	**138**	**0.27**	389	0.77	288	0.57	273	0.54
Time Complexity †	$O(n_{iter} \cdot (n_{nq}+n_{oq}) \cdot (n_{nw}+n_{ow}) \cdot K)$		$O(n_{iter} \cdot n_{nq} \cdot \|\|nq\|\| \cdot K)$		$O(n_{iter} \cdot (n_{nq}+n_{oq}) \cdot (n_{nw}+n_{ow}) \cdot K)$		$O(n_{iter} \cdot n_{nq} \cdot \|\|nq\|\| \cdot K)$		$O(n_{iter} \cdot n_{nq} \cdot \|\|nq\|\| \cdot K)$	

† n_{iter} is the number of iterations, n_{nq} is the number of new questions, n_{oq} is the number of old questions, n_{nw} is the number of new words, n_{ow} is the number of old words and $\|\|nq\|\|$ is the average length of new questions. K is the number of latent topics

Table 4: Model Perplexity comparison among four incremental schemes and batched PLSA on Wenda data set.

Iterations	Batched PLSA	Fold-In	IPLSA	MAP-PLSA	Our method
5	1364.14	1612.15	256561.81	821.86	**419.83**
10	698.11	1575.91	253023.00	695.00	**295.40**
15	555.98	1551.02	251498.60	625.49	**152.47**
20	515.73	1534.14	250698.13	576.60	**112.25**

Table 5: Recommendation precision comparison among four incremental schemes and batched PLSA on Wenda data set.

Iterations	Batched PLSA	Fold-In	IPLSA	MAP-PLSA	Our method
5	40%	33%	20%	35%	**42%**
10	51%	41%	25%	47%	**54%**
15	60%	48%	28%	58%	**63%**
20	71%	59%	32%	61%	**75%**

6.2 Evaluation Metrics

We use three measures to evaluate both the effectiveness and efficiency of the proposed method in comparison with four baseline methods, namely *batched PLSA*, *Fold-In*, *IPLSA*, and *MAP-PLSA*.

The first measure is the *update time*, which indicates how fast the algorithm to update existing model parameters when new data arrives. The measure of *update time* measures the efficiency of the algorithm. To measure the effectiveness of different algorithms, we propose to use two measures: *perplexity* [1] and *precision* [10].

The *perplexity*, widely used in language modeling, is to measure the generalization ability of the model, defined as:

$$Perplexity(D_{test}) = \exp\left\{-\frac{\sum_{i,j} n'(q_i, w_j) \log P(w_j|q_i)}{\sum_{i,j} n'(q_i, w_j)}\right\}, \quad (24)$$

where $P(w_j|q_i)$ is the probability of the word w_j appears in question q_i. $P(w_j|q_i)$ is calculated as follows:

$$P(w_j|q_i) = \sum_{k=1}^{K} P(w_j|z_k)P(z_k|q_i). \quad (25)$$

A lower perplexity score indicates better generalization ability.

Besides the perplexity, the users' judgement of recommendation results is important and essentially the ultimate goal of a recommendation system. The better the recommendation results, the more the users will click on the recommended items. However, it is a subjective task to evaluate users' judgement since it is hard or even impossible to get a well-defined ground-truth. We thus asked 10 volunteers to do manual ratings on different recommendation algorithms. That is, each user is asked to give two rates (relevant *vs.* non-relevant) independently to each of the 20 recommended questions produced by different algorithms. Finally, the precision of the recommendation algorithm is calculated as:

$$Precision = \frac{\#Relevant\ recommendations}{\#All\ recommendations}.$$

There are some hyper-parameters in our algorithm. In our experiments, we set the number of latent topics 64, and the default α is 0.5. For the question or answer that the user posts, the β value in Equation 23 is 0.25. While for the user's positive and negative feedback, the β values are 0.5 and −0.5, respectively. We next present our experiment results.

6.3 Experiment Results

We first show the learned latent topics using the PLSA model. Figure 3 lists six topics from 64 latent topics. Each topic is represented by its top 20 most probable words, *i.e.*, the words are ordered according to $P(w|z)$. We can see that the six latent topics are roughly mapping into six topics, namely *Emotion, Health, Software, Travel, Family*, and *Work*.

Table 3, 4, and Table 5 summarize the comparison results between our algorithm and the four baseline methods in terms of *update time*, *perplexity*, and *precision* respectively.

Table 3 shows the total time spent to incremental update using 500 questions. We can see that batched PLSA is the slowest while Fold-In is the fastest method. This coincides well with our theoretical analysis of time complexity shown in the last column in Table 3. We can also see that it costs 0.54 *second* to process one question on average using our method. This indicates our incremental algorithm is applicable to real-time online systems.

Table 4 shows the perplexity comparison between our algorithm and four baseline methods. The smaller the value of perplexity, the better the performance. We can see that our method achieve the best perplexity results among all the methods. This indicates that our method enjoy a better generalization ability.

Table 5 summarizes the precision of the different algorithms. We

Topic 1 (Emotion)	Topic 2 (Health)	Topic 3 (Software)
离开 一起 幸福 痛	运动 身体 减肥 肌	软件 地址 病毒 瑞
苦 分手 感情 坚	肤 目标 增强 脂	星 安全 浏览器 网
持 祝福 放弃 自	肪 有效 血液 循	页 外观 网站 免
由 付出 伤心 介	环 大腿 仪器 医	费 网址 输入 程
绍 机会 关系 相	生 皮肤 塑造 疗	序 代码 界面 清
爱 希望 心态 表	程 美容 腹部 皮	理 文件 删除 恶
面 借口 心情	下组织 体形 曲线	意 火狐

Topic 4 (Travel)	Topic 5 (Family)	Topic 6 (Work)
丽江 原始 酒店 客	父母 孩子 重要 家	工资 最低 解决 劳
栈 三亚 美丽 草	庭 家里 妈妈 我	动者 职工 教师 报
原 自助 森林 有	们 亲戚 商量 过	酬 税务 支出 住
名 公园 演出 风	年 对方 他家 弟	房 恩格尔 就业 档
俗 金沙 圣地 少数	弟 回家 在家 结	案 奋斗 工厂 人
民族 名胜古迹 景	婚 女友 农村 男	民政府 劳动法 颁
区 冰川 玉龙	生 一辈子	布 规章 全日制

Figure 3: Top words in 6 topics (Ordered by probability).

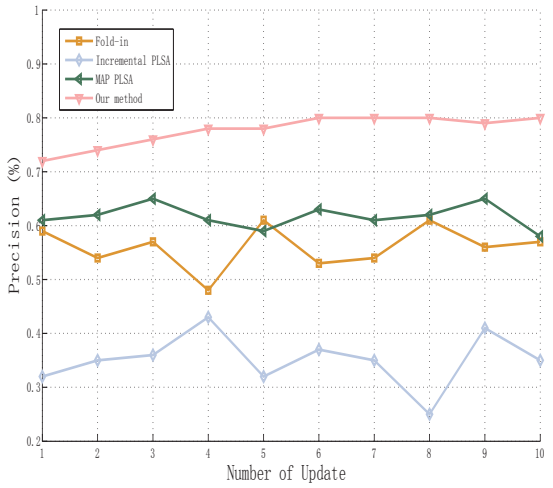

Figure 4: Comparison of recommendation precision of different incremental PLSA algorithms during 10 updates (20 iterations for each update).

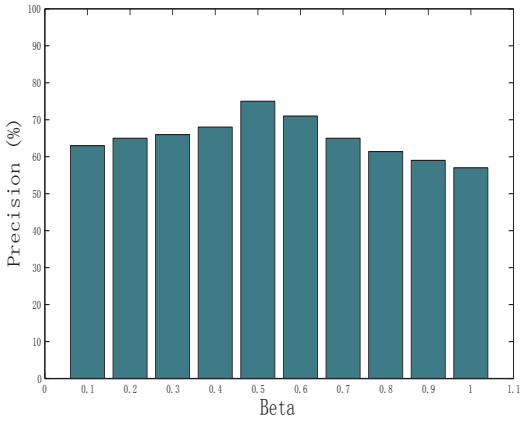

Figure 5: The recommendation precision for different β values after user giving a positive feedback about a recommendation.

can see that our method outperforms the batched PLSA, and other incremental methods. The reason is that our method takes into account uses' short-term interests (setting a bigger weight β) while batched PLSA somehow mixed both users' long-term and short-term interests.

Figure 4 shows the comparison of recommendation results of four incremental algorithms with respect to different number of updates. We can see that the precision of our method improves with the the increase of update numbers, while this trend is not consistent for other methods.

Figure 5 and 6 show the recommendation precision after users giving positive and negative feedback about a recommendation respectively. We can see that the precision can change with the adjustment of the weight β value. In specific, for positive feedback, we set β positive values for better performances, while for negative feedback, we set β negative values to achieve a higher precision. Note that the performance is linearly improved with respect to the value of β. Usually, the value of β is around 0.5, we achieved the best precision.

We illustrate the process of positive and negative feedback with an example shown in Figure 7. In the figure, the system first automatically recommends a list of questions to the user. The questions are roughly about two topics: language study related questions, and finance related questions. Note that the rankings of language and finance related questions are mixed. The user first gives a positive feedback about Question 1 on language. The system updates the recommendation list instantly, and provides an updated list (Figure 7 (ii)). In this list, language related questions are all ranked higher than those of finance related questions. After that, the user gives a negative feedback about a finance related question. Accordingly, our system updates the recommendation list a second time. We can see that all the recommended questions are related to language, and all finance related questions disappear from the recommendations (Figure 7 (iii)).

7. CONCLUSIONS AND FUTURE WORK

In this paper we presented an incremental automatic question recommendation algorithm based on probabilistic latent semantic analysis. The incremental algorithm can update existing model parameters when new data arrives without re-training the whole model from the scratches. Our method can also consider not only users' short-term and long-term interests, but also users' positive and negative feedback. Experiments on a real-world Q & A data set demonstrated both the efficiency and effectiveness of the proposed algorithm.

We have two future work items. First, the proposed incremental algorithm is general as it can be not only applied to other recommendation tasks, but also extendable to other probabilistic latent topic models, such as Latent Dirichlet Allocation (LDA). In future, we plan to extend our work to both new models, such as LDA, and new recommendation applications. Second, it is crucial to evaluate performance of incremental algorithms. We think users' click information is a good indicator of whether he likes (or dislikes) the recommended items. We will use this information for a better evaluation of different recommendation algorithms.

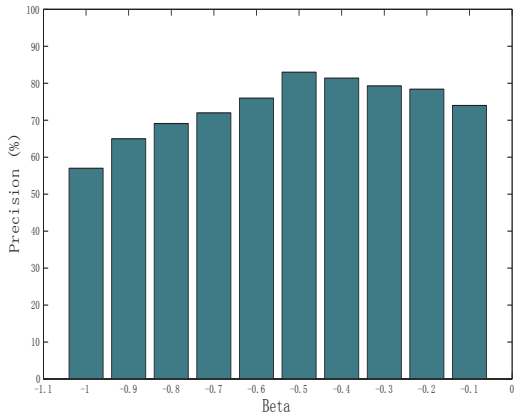

Figure 6: The recommendation precision for different β values after user giving a negative feedback about a recommendation.

8. ACKNOWLEDGEMENT

We would like to thank Sha Huang for preparing the data and Xiance Si for generating some figures.

9. REFERENCES

[1] Thomas Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Maching Learning Journal*, Vol. 42, No. 1-2, pp. 177-196, 2001.

[2] M. Girolami and A. Kaban. On an Equivalence Between PLSI and LDA. In: *Proc. of SIGIR*, pp. 433-434, 2003.

[3] Dempster A. P., Laird N. M., and Rubin D. B.. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, Vol. 39, No. 1, pp. 1-38, 1977.

[4] Christophe G. Carrier. A Note on the Utility of Incremental Learning. *AI Communications*, Vol. 13, No. 4, pp. 215-223, 2000.

[5] T. C. Chou and M.C Chen. Using Incremental PLSA for Threshold Resilient Online Event Anlysis. *IEEE Transaction on Knowledge and Data Engineering*, Vol. 20, No. 3, pp. 289-299, 2008.

[6] T. Hofmann. Latent Semantic Models for Collaborative Filtering. *ACM Transaction Information System*, Vol. 22, No. 1, pp.89-115, 2004.

[7] L. Zhang and C. Li, *etc.*. An Efficient Solution to Factor Drifting Problem in the PLSA Model. In: *Proc. of the The Fifth International Conference on Computer and Information Technology*, pp.175-181, 2005.

[8] J. Zhang, Z. Ghahramani, and Y. Yang. A Probabilistic Model for Online Document Clustering with Applications to Novelty Detection. In *Proc. of NIPS*, pp. 1617-1624, 2005.

[9] Arun C. Surendran and Suvrit Sra. Incremental Aspect Models for Mining Document Streams. *10th European Conferences on Principles and Practice of Knowledge Discovery*, pp. 633-640, 2006.

[10] J. T. Chien and M. S. Wu. Adaptive Bayesian Latent Semantic Analysis. *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 16, No. 1, pp. 198-207, 2008.

[11] B. Marlin. Collaborative Filtering: A Machine Learning Perspective. Master's thesis, University of Toronto, 2004.

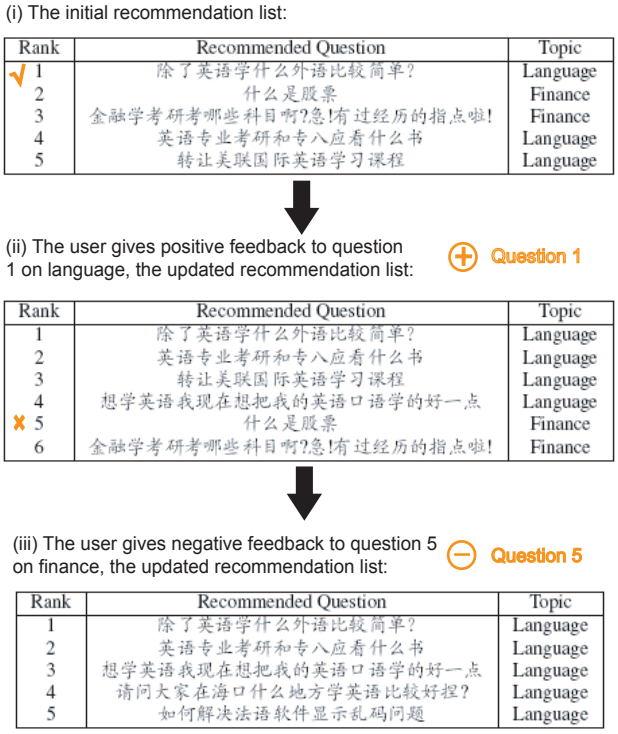

Figure 7: The recommendation list changes after the user giving feedback about the recommended questions.

[12] Das A., Datar M., Garg A. and Rajaram S.. Google News Personalization: Scalable Online Collaborative Filtering. In: *Proc. of the 16th Int. Conf. on World Wide Web*, pp. 270-280, 2007.

[13] D. M. Blei, A. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research* , 3, 993-1022, 2003.

[14] R. M. Neal and G. E. Hinton. A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. In *Learning in Graphical Models. Kluwer Academic Press*, pp. 355-368, 1998.

[15] Arindam Banerjee and Sugato Basu. Topic Models over Text Streams: A Study of Batch and Online Unsupervised Learning. In: *Proc. of the SIAM International Conference on Data Mining (SDM)*, pp.437-442, 2007.

[16] Asela Gunawardana, William Byrne. Convergence Theorems for Generalized Alternating Minimization Procedures. The *Journal of Machine Learning Research*, Vol. 6, pp. 2049-2073, 2005.

[17] Y. B. Cao, H. Z. Duan, C. Y. Lin, Y. Yu, and H. W. Hon. Recommending Questions Using the MDL-based Tree Cut Model. In: *Proc. of the 17th Int. Conf. on World Wide Web*, pp. 81-90, 2008.

[18] Lada A. Adamic, J. Zhang, and *etc.*. Knowledge Sharing and Yahoo Answers: Everyone Knows Something. In: *Proc. of the 17th Int. Conf. on World Wide Web*, pp. 665-674, 2008.

[19] Z. Gyöngyi, G. Koutrika, *etc.*. Questioning Yahoo! Answers. First *WWW Workshop on Question Answering on the Web*, 2008.