Fine-Grained Knowledge Sharing in Collaborative Environments

Ziyu Guan, Shengqi Yang, Huan Sun, Mudhakar Srivatsa, Xifeng Yan

Abstract—In collaborative environments, members may try to acquire similar information on the Web in order to gain knowledge in one domain. For example, in a company several departments may successively need to buy business intelligence software and employees from these departments may have studied online about different business intelligence tools and their features independently. It will be productive to get them connected and share learned knowledge. We investigate fine-grained knowledge sharing in collaborative environments. We propose to analyze members' Web surfing data to summarize the fine-grained knowledge acquired by them. A two-step framework is proposed for mining fine-grained knowledge: (1) Web surfing data is clustered into tasks by a nonparametric generative model; (2) a novel discriminative infinite Hidden Markov Model is developed to mine fine-grained aspects in each task. Finally, the classic expert search method is applied to the mined results to find proper members for knowledge sharing. Experiments on Web surfing data collected from our lab at UCSB and IBM show that the fine-grained aspect mining framework works as expected and outperforms baselines. When it is integrated with expert search, the search accuracy improves significantly, in comparison with applying the classic expert search method directly on Web surfing data.

Index Terms—Advisor search, text mining, Dirichlet processes, graphical models

1 Introduction

Interacting with the Web and with colleagues/friends to acquire information is a daily routine of many human beings. In a collaborative environment, it could be common that members try to acquire similar information on the Web in order to gain specific knowledge in one domain. For example, in a company several departments may successively need to buy business intelligence (BI) software, and employees from these departments may have studied online about different BI tools and their features independently. In a research lab, members are often focused on projects which require similar background knowledge. A researcher may want to solve a data mining problem using nonparametric graphical models which she is not familiar with but have been studied by another researcher before. In these cases, resorting to a right person could be far more efficient than studying by oneself, since people can provide digested information, insights and live interactions, compared to the Web. For the first scenario, it is more productive for an employee to get advices on the choices of BI tools and explanations of their features from experienced employees; for the second scenario, the first researcher could get suggestions on model design and good learning materials from the second researcher. Most people in collaborative environments would be happy to share experiences with and give suggestions to others on specific problems. However, finding a right person is challenging due to the variety of information needs. In this paper, we investigate how to enable such knowledge sharing mechanism by analyzing user data.

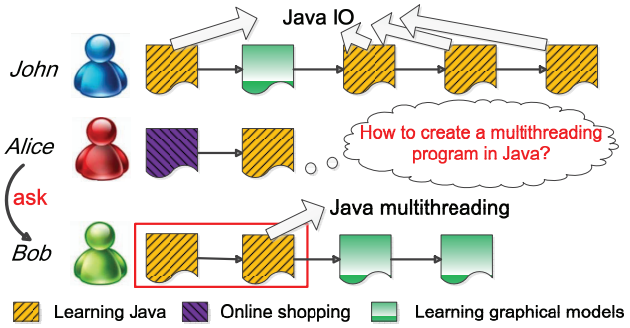


Fig. 1. An illustrative toy example for knowledge sharing in a collaborative environment.

An illustrative toy example is given in Figure 1. One can use "tcpdump" to intercept a sequence of Web surfing activities (IP packets) for each member. The scene is, Alice starts to surf the Web and wants to learn how to develop a Java multithreading program, which has already been studied by Bob (red rectangle). In this case, it might be a good idea to consult Bob, rather than studying by herself. We aim to provide such recommendations by analyzing

- *Z. Guan is with the College of Information and Technology, Northwest University of China, Xi'an, CN 710127. E-mail: welbyhebei@gmail.com*
- *S. Yang, H. Sun and X. Yan are with the Department of Computer Science, University of California, Santa Barbara, CA 93106. E-mail: {sqyang, huansun, xyan}@cs.ucsb.edu*
- *M. Srivatsa is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: msrivats@us.ibm.com*

surfing activities automatically. In this example, not necessarily Bob is an expert in every aspect of Java programming; however, due to his significant surfing activities in Java multithreading, it is reasonable to assume that he has gained enough knowledge in this area so that he can help Alice (in practice we could set a threshold on the amount of related surfing data to test significance). Even if Bob is still learning, he could share his experiences in learning and possibly suggest good learning materials to Alice, thus saving Alice's effort and time.

This scenario departs from the traditional expert search problem in that expert search aims to find domain experts based on their associated documents in an enterprise repository, while our goal is to find proper "advisors" who are most likely possessing the desired piece of fine-grained knowledge based on their Web surfing activities. The semantic structures hidden in Web surfing (as illustrated by Figure 1) reflect people's knowledge acquisition process and make Web surfing data significantly different from enterprise repositories. Traditional expert search methods may not be able to well handle the Web surfing data. For example, in the above example John spent a lot of effort on "Java IO" which is only partially relevant to Alice's need. If we simply treat Web surfing data as a collection of documents and apply traditional expert search methods, John would be ranked higher than Bob since he viewed more contents about "Java", though not quite relevant. We will analyze this issue in detail in Section 6 and demonstrate it empirically in Section 7.2. Therefore, it is necessary to first summarize people's fine-grained knowledge reflected in their Web surfing activities by recognizing the semantic structures, and then search over the mined pieces of fine-grained knowledge (e.g. "Java IO"). We call this search scenario, "advisor search", to differentiate from traditional expert search. We use the term advisor search to emphasize that the knowledge of the retrieved people might not be very deep, but good enough to help others if they have not solidly gained the related knowledge yet.

In order to analyze the knowledge acquired by Web users, we propose to log and analyze users' Web surfing data (not only search, but also browsing activities, which reveal a user's knowledge gaining process). Users' interactions with the Web can be segmented into different "tasks", e.g., "learning Java" and "shopping". Textual contents of a task are usually cohesive. We define a *session* (a document in Figure 1) as an aggregation of consecutively browsed Web contents of a user that belong to the same task. Sessions are atomic units in our analysis. The content of sessions in a task could evolve gradually: people usually learn basic concepts first and then move towards advanced topics. A task can be further decomposed into fine-grained aspects (called *micro-aspects*). A micro-aspect could be roughly defined as a significantly more cohesive subset of sessions in a task. For example, the task "learning Java" might contain "Java IO" and "Java multithreading" as two micro-aspects. When pursuing a task, a user could spend many sessions on a micro-aspect. Mining these micro-aspects (micro-knowledge) is critical: it can provide a detailed description of the knowledge gained by a person, which is the basis for advisor search.

We propose a two-step framework for mining fine-grained knowledge (micro-aspects): (1) In the first step, we formulate tasks from sessions. We design an infinite Gaussian mixture model based on Dirichlet Process (DP) [12] to cluster sessions. We adopt a nonparametric scheme since the number of tasks is difficult to predict. (2) We then extract micro-aspects from sessions in each task. The challenges are: the number of micro-aspects in a task is unknown; sessions for different micro-aspects of a task are textually similar; sessions for a micro-aspect might not be consecutive. To this end, a novel discriminative infinite Hidden Markov Model (d-iHMM) is proposed to mine micro-aspects and evolution patterns (if any) in each task. A background model is introduced in order to enhance the discriminative power. Finally, we apply a language model based expert search method [1] over the mined micro-aspects for advisor search.

To our knowledge, there is no existing techniques for micro-aspect mining. Although the hierarchical topic modeling algorithm [5] can discover general-to-specific topic hierarchies, it decomposes sessions into topics but not groups them. A person with many sessions containing partially relevant topics would still be ranked unexpectedly high (like the "John vs. Bob" problem aforementioned). Our goal is to detect people's online learning activities (e.g. learning "Java IO") in session data reflected by subsets of sessions, rather than discerning topics hierarchically in sessions (e.g. "Java" with "IO" as its subtopic). Mining the semantic structures in sessions (Figure 1) is important for advisor search, as will be shown in Section 6.

The proposed two-step framework intrinsically groups sessions into micro-aspects in a coarse-to-fine fashion, which bears some similarity to hierarchical clustering [30]. However, traditional hierarchical clustering methods could not handle session data well. This is because people usually go to the same Website for different micro-aspects of a task. For example, researchers could commonly use Google Scholar[1] to look for papers related to *Clustering*. Two micro-aspects, "spectral clustering" and "density-based clustering", can be difficult to separate since there are a lot of *background contents* in their sessions, e.g. navigational texts, template texts, etc. These background contents can drastically blur the boundary between the two micro-aspects, considering they are already similar. Hence, traditional hierarchical clustering methods

1. http://scholar.google.com

could easily mess up micro-aspects of a task, while the proposed d-iHMM model can better separate different micro-aspects since it models the background contents explicitly. The experimental results verify our analysis: d-iHMM does better than iHMM [2] (which can be approximately regarded as clustering sessions into micro-aspects without background modeling) in micro-aspect mining.

The contributions of this work are summarized as follows. (1) We propose the fine-grained knowledge sharing problem in collaborative environments. The goal is not finding domain experts but a person who has the desired specific knowledge. This problem is significant in practice in that learning from an advisor (if she/he is easy to find) might be more efficient than studying on the Web (though not always). A lot of repeating efforts could be saved. (2) We propose to solve this problem by first summarizing Web surfing data into fine-grained aspects, and then search over these aspects. We compare this strategy with searching advisors directly over sessions (i.e. applying traditional expert search methods on Web surfing data directly) both analytically (Section 6) and empirically (Section 7.2). (3) We propose a novel two-step micro-aspect mining framework consisting of two nonparametric DP models. This framework does not require pre-specified number of tasks and number of micro-aspects of a task, and can correctly summarize sessions into micro-aspects by explicitly modeling background contents in sessions of a task. (4) We collect real user-generated Web surfing data at our lab and at IBM to test the feasibility of our idea. Experiments on both datasets show that our scheme is effective and outperforms the one using raw session data. We also show the proposed DP models work as expected and achieve better performance than baseline methods. Certainly, the selection of these algorithms is not unique. Through this study, we demonstrate the possibility of finding right persons automatically by analyzing their Web surfing data.

The rest of the paper is organized as follows: the next section outlines related work. Section 3 gives the problem formulation. In Section 4 we present the Gaussian DP model for clustering sessions into tasks. Section 5 describes the proposed d-iHMM model for mining fine-grained aspects in each task (i.e. session cluster), followed by a discussion of the advisor search method based on the mined micro-aspects and a detailed analytical comparison to searching directly over sessions in Section 6. Experiments are described in Section 7, and finally, Section 8 concludes our work.

2 Related Work

In this section we review research fields that are related to our work: expert search, analysis of user search tasks and topic modeling.

2.1 Expert Search

Expert search aims at retrieving people who have expertise on the given query topic. Early approaches involve building a knowledge base which contains the descriptions of people's skills within an organization [8]. Expert search became a hot research area since the start of the TREC enterprise track [9] in 2005. Balog *et al.* proposed a language model framework [1] for expert search. Their Model 2 is a document-centric approach which first computes the relevance of documents to a query and then accumulates for each candidate the relevance scores of the documents that are associated with the candidate. This process was formulated in a generative probabilistic model. Balog *et al.* showed that Model 2 performed better [1] and it became one of the most prominent methods for expert search. Other methods have been proposed for enterprise expert search (e.g. [11], [24]), but the nature of these methods is still accumulating relevance scores of associated documents to candidates. Expert retrieval in other scenarios has also been studied, e.g. online question answering communities [19], academic society [10].

The proposed advisor search problem is different from traditional expert search. (1) Advisor search is dedicated to retrieving people who are most likely possessing the desired piece of fine-grained knowledge, while traditional expert search does not explicitly take this goal. (2) The critical difference lies in the data, i.e. sessions are significantly different from documents in enterprise repositories. A person typically generates multiple sessions for a micro-aspect of a task, e.g. a person could spend many sessions learning about Java multithreading skills. In other words, *the uniqueness of sessions is that they contain semantic structures which reflect people's knowledge acquisition process*. If we treat sessions as documents in an enterprise repository and apply the traditional expert search methods (e.g. [1]), we could get incorrect ranking: due to the accumulation nature of traditional methods, a candidate who generated a lot of marginally relevant sessions (same task but other micro-aspects) will be ranked higher than the one who generated less but highly relevant sessions, e.g. John vs. Bob in Figure 1 for the query "Java multi-thread programming" (Section 6 will provide a detailed analysis of this issue). Therefore, it is important to recognize the semantic structures and summarize the session data into micro-aspects so that we can find the desired advisor accurately. In this paper we develop nonparametric generative models to mine micro-aspects and show the superiority of our search scheme over the simple idea of applying traditional expert search methods on session data directly.

2.2 Analysis of Search Tasks

Recently, researchers have focused on detecting, modeling and analyzing user search tasks from query logs. Here we name some representative works. Jones and Klinkner found that search tasks are interleaved and used classifiers to segment the sequence of user queries into tasks [15]. Liu and Belkin combined task stage and task type with dwell time to predict the usefulness of a result document, using a 3-stage and 2-type controlled experiment [18]. Ji *et al.* used graph regularization to identify search tasks in query logs [14]. Kotov *et al.* designed classifiers to identify same-task queries for a given query and to predict whether a user will resume a task [16]. Wang *et al.* formulated the cross-session search task mining problem as a semi-supervised clustering problem where the dependency structure among queries in a search task was explicitly modeled and a set of automatic annotation rules were proposed as weak supervision [28].

This line of research tries to recover tasks from people's search behaviors and bears some similarity to our work. Nevertheless, our work differs from theirs from the following aspects. First, we consider general Web surfing contents (including search), rather than search engine query logs. Query logs do not record the subsequent surfing activity after the user clicked a relevant search result. Moreover, it is found that 50% of a user's online pageviews are content browsing [17]. Web surfing data provides more comprehensive information about the knowledge gaining activities of users. Although various methods were proposed for extracting search tasks in query logs, these methods cannot be applied in our setting since they exploit query log specific properties. Second, none of the above works tried to mine fine-grained aspects for each task. When studying, people could spend some effort on one fine-grained aspect of a task and generate multiple contents. Summarizing fine-grained aspects can provide a fine-grained description of the knowledge gained by a person. Finally, none of existing works which analyze user online behaviors (not limited to search behaviors, e.g. [29]) tried to address advisor search by exploiting the data generated from users' past online behaviors.

2.3 Topic Modeling

Topic modeling is a popular tool for analyzing topics in a document collection. The most prevalent topic modeling method is Latent Dirichlet Allocation (LDA) [7]. Based on LDA, various topic modeling methods have been proposed, e.g. the dynamic topic model for sequential data [6] and the hierarchical topic model for building topic hierarchies [5]. The Hierarchical DP (HDP) model can also be instantiated as a nonparametric version of LDA [25]. However, our problem is not a topic modeling problem. Our goal is to recover the semantic structures of people's online learning activities from their Web surfing data, i.e. identifying groups of sessions representing *tasks* (e.g. learning "Java") and *micro-aspects* (e.g. learning "Java multithreading"). While topic modeling decomposes a document into topics. After applying topic modeling methods on session data, it is still difficult to find the right advisor by using the mined topics. This is because a person with many sessions containing partially relevant topics would still be ranked unexpectedly high, due to the accumulation of relevance among sessions. Grouping sessions into micro-aspects is important for advisor search.

3 Problem Formulation

We refer to a member in a collaborative environment as a "candidate". In our problem, we have a group of h candidates $\{e_1, \ldots, e_h\}$ where each candidate e_i generates a sequence of sessions $\mathcal{W}^{(i)} = \{w_1^{(i)}, \ldots, w_{N_i}^{(i)}\}$. For clarity, we group sessions of different candidates together using a uniform subscript index: $\mathcal{W} = \{w_1, \ldots, w_N\}$, with totally N sessions. The fine-grained knowledge sharing problem consists of three subproblems:

1) Partition $\mathcal{W}$ into a set of clusters $\mathcal{C} = \{C_1, \ldots, C_t\}$ where each cluster represents a task;
2) Partition sessions in each C_i into a set of micro-aspects $\mathcal{S}_i = \{s_{i1}, \ldots, s_{it_i}\}$, where each micro-aspect s_{ij} is a significantly more cohesive subset of sessions from C_i.
3) Compute the association weight between e_i and s_{jk} as $|\mathcal{W}^{(i)} \bigcap s_{jk}|$. Given a query q, produce a ranking of $\{e_i\}$ according to their relevance to q assessed by the relevance of the associated micro-asepcts and the corresponding association weights.

4 Session Clustering

The input of this step is $\mathcal{W}$, where each w_i is a $D_0 \times 1$ word frequency vector with D_0 as the vocabulary size. The intuition is that contents generated for the same task are textually similar while those for different tasks are dissimilar. Hence, clustering is a natural choice for recovering tasks from sessions. In our case, it is difficult to preset the number of tasks given a collection of sessions. Therefore, we need to automatically determine the number of clusters (k), which is also one of the most difficult problems in clustering research. Most methods for automatically determining k run the clustering algorithm with different values of k and choose the best one according to a predefined criterion [13], which could be costly. In Spectral Clustering, a heuristic for determining k is to search for a significant raise in the magnitude of the eigenvalues [27]. However, this does not work in our context since the Web contents are so noisy that

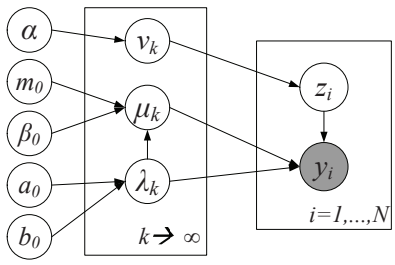


Fig. 2. The graphical representation of the Gaussian Dirichlet Process mixture model.

eigenvalues start to raise gradually from the second smallest eigenvalue. In this work, we advocate using a generative model with a Dirichlet Process (DP) prior [12] for clustering. DPs provide nonparametric priors for k and the most likely k is learned automatically. A DP, written as $G \sim \mathrm{DP}(\alpha, G_0)$, can be interpreted as drawing components (clusters here) from an infinite component pool, with α called the scaling parameter and G_0 being the prior for a random component. An intuitive interpretation of DP is the stick-breaking construction:

$$\pi_i(v) = v_i \prod_{j=1}^{i-1}(1 - v_j), \qquad G = \sum_{i=1}^{\infty} \pi_i \delta_{\psi_i},$$

where $v = \{v_1, v_2, \dots\}$ with each v_i drawn from the Beta distribution $\mathrm{Beta}(1, \alpha)$, ψ_i is a component drawn from G_0 and δ_{ψ_i} is an atom at ψ_i. π_i is the mixture weight of ψ_i given by breaking the current length of the "stick" (i.e. $\prod_{j=1}^{i-1}(1 - v_j)$) by the fraction v_i. The generation of π is often written as $\pi \sim \mathrm{GEM}(\alpha)$. π defines a prior mixing distribution among the infinite many components. The posterior mixing distribution and the real number of components drawn from the DP is then learned from the data. Readers are referred to [25] for a detailed description of DPs.

4.1 Clustering by GDP Mixture Model

When using probabilistic models for clustering, the Gaussian mixture model is a common choice and can be viewed as a probabilistic version of k-means [13]. However, the data dimensionality D_0 is too high to apply Gaussian distributions in our case (often above 10K). Therefore, we first apply the well-known Laplacian Eigenmap (LE) technique [3] to reduce the dimensionality from D_0 to D where $D_0 \gg D$. We choose LE since it could also capture the nonlinear manifold structure of a task, e.g. the topics of a task could evolve and drift which could be described by the "half-moon" structure [3].

Let $\mathcal{Y} = \{y_1, \dots, y_N\}$ denote the session vectors in the subspace learned by LE. The graphical representation of GDP is depicted in Figure 2. The DP prior is represented by the stick-breaking construction process, with $\{v_k\}_{k=1}^{\infty}$ and α defined above. z_i is an assignment variable of the mixture component (i.e. cluster) with which y_i is associated. μ_k and λ_k are $D \times 1$ vectors denoting the mean vector and the diagonal precision matrix's diagonal vector of the k-th Gaussian component. This means each dimension of the data is independent. $\{m_0, \beta_0, a_0, b_0\}$ are the set of hyperparameters for the Gaussian-Gamma conjugate prior of Gaussian components. The generative process of GDP is as follows

1) Draw $v_k|\alpha \sim \mathrm{Beta}(1, \alpha)$, $k = 1, 2, \dots$
2) Draw $\lambda_{kj}|a_0, b_0 \sim \mathrm{Gamma}(a_0, b_0)$, $k = 1, 2, \dots; j = 1, \dots, D$
3) Draw $\mu_k|m_0, \beta_0, \lambda_k \sim \mathcal{N}(m_0, (\beta_0 \mathbf{D}^{\lambda_k})^{-1})$, $k = 1, 2, \dots$
4) For the i-th session:
   a) Draw $z_i|v \sim \mathrm{Mult}(\pi(v))$
   b) Draw $y_i|z_i, \mu, \lambda \sim \mathcal{N}(\mu_{z_i}, (\mathbf{D}^{\lambda_{z_i}})^{-1})$

Here $\mathrm{Mult}(\cdot)$ denotes the multinomial distribution. $\mathbf{D}^{\lambda_k}$ is a diagonal matrix with elements of λ_k on its diagonal.

The infinite Gaussian mixture model has already been proposed and used for clustering. The overall design of our model follows previous work, but with a different design choice for the prior of λ_k. Previous models treat λ_k either as the full precision matrix with a Wishart prior [23] or as a scalar with a Gamma prior [21]. The former design choice is able to model complicated cluster structures but the time cost is high, while the modeling power of the latter one is very limited. In our model, λ_k is a vector, meaning that different dimensions are independent and have different precisions. The reasons are: (1) the output of LE is the orthogonal eigenvectors of a real symmetric matrix, which means different dimensions are independent; (2) different dimensions may have different degrees of variation. In this way, the model is relatively more efficient and, meanwhile, retains certain expressive power.

Either Gibbs sampling or variational inference can be used to solve the GDP model. Although Gibbs sampling can provide theoretical guarantees of accuracy, variational inference converges much faster and can also provide a reasonable approximation to the real posteriors [4]. Hence, we choose variational inference in this work. The derivation is simply an application of the general derivation in [4]. Readers can refer to the appendix (as a supplemental material) for a detailed description of the variational inference of the GDP model.

5 Mining Fine-Grained Knowledge

The major challenge of mining micro-aspects is that the micro-aspects in a task are already similar with one another. If we model each component (i.e. micro-aspect) independently (as most traditional models do), it is likely that we mess up sessions from different micro-aspects, i.e. leading to bad discrimina-

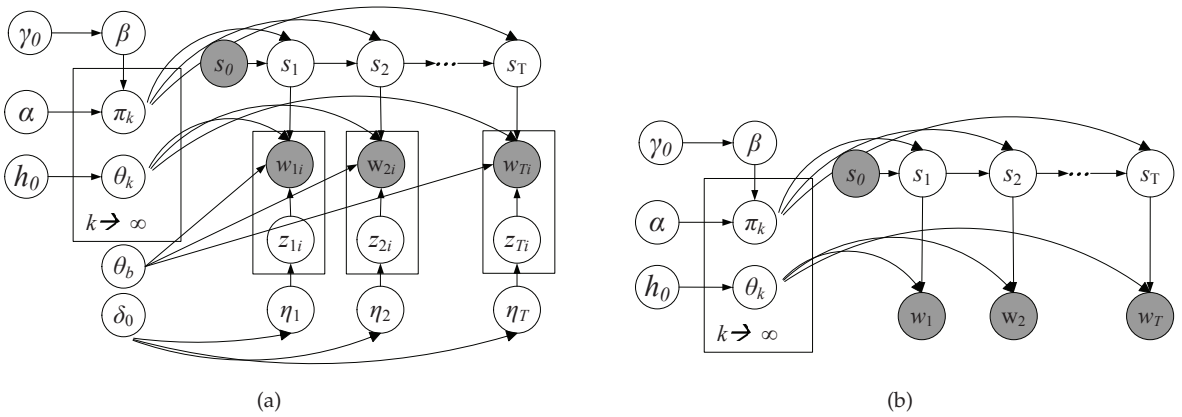


Fig. 3. The graphical representations of (a) the d-iHMM model, and (b) the original iHMM model.

tion. Therefore, we should model different micro-aspects in a task jointly, separating the common content characteristics of the task from the distinctive characteristics of each micro-aspect. To this end, we extends the infinite Hidden Markov Model (iHMM) [2] and propose a novel discriminative infinite Hidden Markov Model (d-iHMM) to mine micro-aspects and possible evolution patterns in a task. The graphical representation is shown in Figure 3(a). The observed variables of the model is a sequence of T sessions $\{w_t\}_{t=1}^{T}$ belonging to a cluster outputted by the GDP. In practice, a cluster can contain multiple sequences from different people. However, we only discuss the single sequence case for clarity. Extension to the multi-sequence case is a trivial work since those sequences do not depend on each other. A background unigram model, θ_b, which is estimated by aggregating all the task's sessions, is shared among states (i.e. micro-aspects). s_t's are state assignment variables. z_{ti} is an indicator variable, controlling whether word w_{ti} is generated by θ_{s_t} ($z_{ti} = 1$) or θ_b ($z_{ti} = 0$). $\eta_t \in (0, 1)$ encodes the fraction of words in w_t which are not generated by θ_b, with δ_0 as a Beta prior. In this way, the common background contents in the task's sessions are explained by θ_b, thus increasing the discriminative power of the model. d-iHMM is fundamentally based on the Hierarchical DP model [25] where the infinite component pools (corresponding to $\{\pi_k\}$) of all states share the same base infinite component pool (corresponding to β). The generative process is summarized as follows

1) Draw $\beta|\gamma_0 \sim \mathrm{GEM}(\gamma_0)$
2) For $k = 1, 2, \ldots$, draw $\pi_k \sim \mathrm{DP}(\alpha, \beta)$, $\theta_k \sim \mathrm{Dir}(h_0, \ldots, h_0)$
3) For $t = 1$ to T:
   a) Draw $s_t|s_{t-1} \sim \mathrm{Mult}(\pi_{s_{t-1}})$, $\eta_t|\delta_0 \sim \mathrm{Beta}(1, \delta_0)$
   b) For each word w_{ti} in the t-th session
      i) Draw $z_{ti}|\eta_t \sim \mathrm{Bernoulli}(\eta_t)$
      ii) If $z_{ti} = 1$, draw $w_{ti} \sim \mathrm{Mult}(\theta_{s_t})$, else draw $w_{ti} \sim \mathrm{Mult}(\theta_b)$

Here $\mathrm{Dir}(h_0, \ldots, h_0)$ is a symmetric Dirichlet distribution where the dimension is the vocabulary size, D_0. $p(s_1|s_0)$ is given by a uniform multinomial distribution. As in [25], we place Gamma priors on the hyperparameters α and γ_0: $\alpha \sim \mathrm{Gamma}(a_\alpha, b_\alpha)$ and $\gamma_0 \sim \mathrm{Gamma}(a_{\gamma_0}, b_{\gamma_0})$. The evolution patterns learned (if statistically strong) could be used for content recommendation. Furthermore, d-iHMM could be iteratively applied on the learned micro-aspects to find more fine-grained aspects.

The key difference between d-iHMM and iHMM is the background model part, i.e., θ_b, δ_0, η_t's and z_{ti}'s are all new nodes compared to the graphical model of iHMM (Figure 3(b)). Modeling the common background contents makes d-iHMM significantly different from iHMM and enhances the discriminative power so that we can identify relatively more cohesive subsets of documents from a set of cohesive documents with background contents.

5.1 Solving d-iHMM by Beam Sampling

The beam sampling method for iHMM is proposed in [26], which is shown to converge to the true posterior much faster than a classical Gibbs sampler. Therefore, we develop a beam sampler for our d-iHMM model.

Beam sampling adopts the slice sampling [22] idea to limit the number of states considered at each time step to a finite number, so that dynamic programming can be used to sample whole state trajectories efficiently. Specifically, we first sample an auxiliary variable u_t for each time step t in the sequence with conditional distribution $u_t \sim \mathrm{Uniform}(0, \pi_{s_{t-1}s_t})$. Given $\{u_t\}$, the sequence $\{s_t\}$ is re-sampled, considering only the values of s_{t-1} that satisfy $\pi_{s_{t-1}s_t} > u_t$. Hence, $\{u_t\}$ act as a truncation of the infinite number of states and make the number of trajectories with positive probabilities finite, so that the whole sequence can be sampled holistically. $\{s_t\}$ is sampled in one run by a forward filtering backward sampling algorithm.

Each sampling iteration samples $\{u_t\}$, $\{s_t\}$, $\{z_{ti}\}$, $\{\theta_k\}$, $\{\pi_k\}$ and β in turn. We first sample $\{u_t\}$ as described above and create more states if the maximum unassigned probabilities in $\{\pi_k\}$ (the last element of each π_k) is greater than the minimum of $\{u_t\}$. Then we perform a forward sweep of $\{s_t\}$ where for the t-th step we compute

$$\begin{aligned} & p(s_t|w_{1:t}, u_{1:t}, z_t) \quad (1)\\ \propto & p(s_t, u_t, w_t|w_{1:t-1}, u_{1:t-1}, z_t)\\ = & p(w_t|s_t, z_t)\\ & \times \sum_{s_{t-1}} \mathbb{I}(u_t < \pi_{s_{t-1}s_t}) p(s_{t-1}|w_{1:t-1}, u_{1:t-1}, z_{t-1}), \end{aligned}$$

where $p(w_t|s_t, z_t)$ means we generate a word w_{ti} by θ_{s_t} only if $z_{ti} = 1$, and $\mathbb{I}(C) = 1$ if condition C is true and 0 otherwise. s_T is sampled from $p(s_T|w_{1:T}, u_{1:T}, z_T)$. Then we perform a backward sweep to sample each s_t given s_{t+1} by

$$\begin{aligned} & p(s_t|s_{t+1}, w_{1:T}, u_{1:T}, z_t) = p(s_t|s_{t+1}, w_{1:t}, u_{1:t+1}, z_t) \quad (2)\\ \propto & p(s_t|w_{1:t}, u_{1:t+1}, z_t) p(s_{t+1}|s_t, u_{t+1})\\ \propto & p(u_{t+1}|s_t) p(s_t|w_{1:t}, u_{1:t}, z_t) \frac{p(u_{t+1}|s_t, s_{t+1}) p(s_{t+1}|s_t)}{p(u_{t+1}|s_t)}\\ = & p(s_t|w_{1:t}, u_{1:t}, z_t) \mathbb{I}(u_{t+1} < \pi_{s_t s_{t+1}}) \end{aligned}$$

In order to efficiently sample z_{ti}, we integrate out $\{\eta_t\}$. This makes all z_{ti}'s belonging to w_t dependent on one another. Let $z_{\neg ti}$ be the set of z variables for w_t except z_{ti}. We have

$$\begin{aligned} & p(z_{ti}|z_{\neg ti}, \delta_0, w_t, \theta, \theta_b, s_t) \quad (3)\\ \propto & p(z_{ti}, z_{\neg ti}, w_t|\delta_0, \theta, \theta_b, s_t) = p(z_t|\delta_0) p(w_t|z_t, s_t, \theta, \theta_b)\\ \propto & \frac{\mathrm{B}(\sum_j z_{tj} + 1, |w_t| - \sum_j z_{tj} + \delta_0)}{\mathrm{B}(1, \delta_0)} p(w_{ti}|z_{ti}, s_t, \theta, \theta_b), \end{aligned}$$

where $|w_t|$ is the number of words in w_t. The final sampling probability ratio is (omitting variables in the condition for clarity):

$$\frac{p(z_{ti} = 1|\cdots)}{p(z_{ti} = 0|\cdots)} = \frac{p(w_{ti}|\theta_{s_t})(\sum_{j \neq i} z_{tj} + 1)}{p(w_{ti}|\theta_b)(|w_t| - \sum_{j \neq i} z_{tj} + \delta_0 - 1)}, \quad (4)$$

where $p(w|\theta_k)$ means the probability of generating word w by θ_k.

For θ_k, since the prior Dirichlet distribution is conjugate to the multinomial distribution, the posterior sampling distribution of θ_k is

$$\mathrm{Dir}(h_0 + w(k,1), h_0 + w(k,2), \ldots, h_0 + w(k,d)), \quad (5)$$

where $w(k,i)$ is the total number of word occurrences of the i-th word in the vocabulary which are generated by θ_k.

The sampling distributions of π_k and β follow directly from [25], but we briefly describe them for completeness. Let n_{ij} be the number of times we jump from state i to state j. Let M be the number of distinct states in the sequence, relabeled from 1 to M. Merging the infinitely many states not represented in the sequence into one state, the sampling distribution of $(\pi_{k1}, \ldots, \pi_{kM}, \sum_{k'=M+1}^{\infty} \pi_{kk'})$ is

$$\mathrm{Dir}(\alpha\beta_1 + n_{k1}, \ldots, \alpha\beta_M + n_{kM}, \alpha \sum_{i=M+1}^{\infty} \beta_i). \quad (6)$$

To sample β, a set of auxiliary variables $\{m_{ij}\}$ is used with independent conditional distributions

$$p(m_{ij} = m|s, \beta, \alpha) \propto S(n_{ij}, m)(\alpha\beta_j)^m, \quad (7)$$

where $S(\cdot,\cdot)$ denotes Stirling numbers of the first kind. $(\beta_1, \ldots, \beta_M, \sum_{k'=M+1}^{\infty} \beta_{k'})$ is then sampled by $\mathrm{Dir}(m_{\cdot 1}, \ldots, m_{\cdot M}, \gamma_0)$ with $m_{\cdot k} = \sum_{k'=1}^{M} m_{k'k}$.

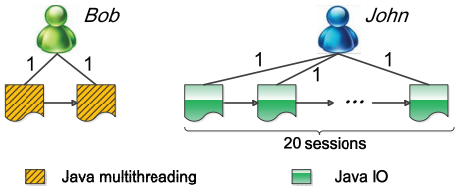


Fig. 4. A toy example for comparing session based and micro-aspect based advisor search.

6 Advisor Search

After we obtain the mined micro-aspects of each task, advisor search can then be implemented on the collection of learned micro-aspects. We employ the traditional language model based expert search method [1]. Let d be a document (i.e. micro-aspect). Given a query q, the method uses $p(e|q)$ to rank advisor candidates. By assuming uniform prior distributions $p(e)$ and $p(d)$ and applying Bayes' rule, it is equivalent to rank candidates by $p(q|e) = \sum_d p(q|d)p(d|e)$ or $p(q|e) \propto \sum_d p(q|d)p(e|d)$ [1]. $p(q|d)$ is the probability of generating q by d's unigram model, with proper smoothing [1]. Intrinsically, the method can be viewed as a weighted accumulation of $p(q|d)$'s from the associated documents of e. Recall that the weight between e and d is the number of sessions of e which fall in d. $p(d|e)$ and $p(e|d)$ encode the normalized association weights between candidates and documents from a candidate's perspective (*candidate scheme*) and a document's perspective (*document scheme*), respectively. The candidate scheme is not intuitive in our context. Consider two candidates e_1 and e_2. e_1 viewed totally 100 sessions in which 10 sessions fall in d, while for e_2 the two numbers are 10 and 2. Hence, $p(d|e_1) = 0.1 < p(d|e_2) = 0.2$. However, e_1 viewed more sessions in d than e_2 and should have a stronger association. Therefore, the document scheme is used for ranking.

Compared to applying traditional expert search methods directly on session data, searching over micro-aspects has the advantage that the associations

between candidates and "documents" are correctly normalized. A toy example is shown in Figure 4. Suppose a user wants to query about "Java multi-thread programming". Suppose orange (with black lines) sessions' relevance scores are 0.9 and those of green sessions are 0.1. Note the association weights between candidates and sessions are all 1. If we apply the language model method on sessions directly (which means sessions are "documents"), the ranking scores of Bob and John are 1.8 and 2 respectively[2]. In contrast, by summarizing the two micro-aspects in Figure 4 and employing micro-aspects as "documents", Bob's score becomes 0.9 while that of John becomes 0.1, which is our expectation.

Time Complexity The time cost of GDP can be regarded roughly as $O(NKDI)$ where I is the number of iterations and K is the truncation level of the variational distribution for the infinite Gaussian mixtures (see appendix). If other variables are fixed, the time cost is approximately linear in N. The time cost of d-iHMM is $O((M^2(T+1)+2(M+T)+N_w)I)$ where I is the number of sampling iterations, M is the number of states (could vary in different iterations) and N_w is the total number of words in the T sessions. d-iHMM could be more costly than GDP given $T = N$, since N_w can be much larger than T. Fortunately, the runs of d-iHMM on different tasks are independent and consequently can be parallelized. It is possible to further speed up GDP and d-iHMM by incorporating recent parallel computing techniques for machine learning algorithms (e.g. [20]). Nevertheless, parallel computing is beyond the focus of this work, which is to demonstrate the feasibility of summarizing fine-grained aspects from Web surfing data for advisor search. The advisor search step is efficient since the main cost is only one matrix vector multiplication ($O(S_a h)$, where S_a is the total number of micro-aspects and h is the number of candidates).

7 Experiments

We validate our methods for fine-grained knowledge mining and advisor search on real Web surfing data. Firstly, we show the results of advisor search using the full pipeline. Then we evaluate the individual models, GDP for session clustering and d-iHMM for fine-grained knowledge mining.

7.1 Datasets

We collected Web surfing data from two real collaborative environments: (1) the data mining lab at UCSB, and (2) the "networking" research group in IBM T. J. Watson Research Center. The first dataset, *Surf-Lab*, consists of surfing data from 8 students in the data mining lab at UCSB. We run the "tcpdump" program ("windump" in windows) on each student's PC, in order to record their surfing activities. Afterwards, HTTP packets (only those generated by Web browers) and the corresponding textual contents were extracted from the dump files. The second dataset, Surf-IBM, is collected from the gateway at IBM T. J. Watson Research Center. Web surfing packets generated by 20 research scientists in the "networking" research group are captured. The Surf-IBM dataset is processed in a similar way. Both datasets span nearly two months. The sequence of http contents of each person was segmented into sessions according to the following rule: we place a session boundary between two consecutive contents if their timestamps are at least 10 minutes away from each other, or their cosine similarity is below a threshold (0.5 in this work). Surf-Lab has 686 sessions and Surf-IBM has 3,925 sessions.

TABLE 1
Queries for Surf-IBM (delimited by ";").

Disruption tolerant network protocols; DoS attack; Intrusion detection vulnerability scanning; Cloud computing bigtable; Database data mirroring; Big data Big Blue; IBM websphere; Network topology ring; Network collapsed backbone; Mobile location network protocols; Virus malware; 802.11b protocol; Python programming; Machine learning jeopardy; Gateway antivirus program; Network traffic congestion; Security information Management authorization; DB2 database; Database data integrity; Network virtualization

7.2 Advisor Search

We first show the results of advisor search. Three schemes are compared: session-based, micro-aspect-based and task-based, with an increasing granularity. The language model based expert search method mentioned in Section 6 is used as the retrieval method. We have tried using other traditional expert search methods, but the results are very similar since they all intrinsically accumulate relevance scores of associated "documents" to candidates. For each scheme, a language model is constructed for each "document", i.e. a session, a micro-aspect, or a task, by aggregating all the texts belonging to it. Note that the session-based scheme is intrinsically applying the traditional language model based expert search method on Web surfing data directly.

The evaluation methodology is as follows: we generate 20 queries for each dataset. These queries represent the fine-grained knowledge required by the specific projects on which the candidates were working

2. We also use the document scheme here. The reason is similar: if we use the candidate scheme, e_1 who viewed 10 sessions about "Java IO" in total 100 sessions may receive a lower score than e_2 who viewed 2 in total 10 sessions for a query q related to "Java IO". Assuming the relevance scores of "Java IO" sessions are all 0.9, it is easy to verify that, using the candidate scheme, $p(q|e_1) = \sum_{i=1}^{10} 0.9 \times 0.01 = 0.09$ and $p(q|e_2) = \sum_{i=1}^{2} 0.9 \times 0.1 = 0.18$

during the data collection period. Queries for Surf-IBM are shown in Table 1 as some examples. The ground truth labels of a query are obtained by showing candidates their top relevant sessions assessed by the language model method and asking them to assign a relevance score to themselves for the data collection period on a scale from 0 to 2: (1) score=0 means "irrelevant"; (2) score=1 means "partially relevant", i.e. he/she has background knowledge for the query; (3) score=2 means "very relevant". Normalized Discount Cumulative Gain (NDCG) is used as the evaluation metric:

$$\text{NDCG@n} = Z_n \sum_{i=1}^{n} \frac{2^{r_i} - 1}{\log_2(i+1)}, \tag{8}$$

where r_i is the relevance score of the candidate at rank i and Z_n is a normalization term to let the perfect ranking have a NDCG value of 1. We focus on $n = 1, 2, 3$, since it is costly for a user to verify many suggested advisors. It is important to place correct people at highest positions.

The results are shown in Tables 2 and 3. We can see that the micro-aspect-based scheme outperforms the other two schemes. One-tailed Wilcoxon test is performed based on all 40 queries to evaluate the significance of the superiority. In particular, the micro-aspect-based scheme is significantly better than the session-based scheme at significance level 0.05, and significantly better than the task-based scheme at significance level 0.01. The task-based scheme is the worst (significantly worse than the session-based scheme at significance level 0.05). This is because a task can contain multiple micro-aspects which might not match a user's need very well.

TABLE 2
Comparison of three advisor search schemes on the Surf-Lab dataset. The session-based scheme is intrinsically applying the traditional language model based expert search method on Web surfing data directly.

Method	NDCG@1	NDCG@2	NDCG@3
Micro-aspect-based	**.883**	**.918**	**.942**
Session-based	.783	.839	.885
Task-based	.783	.825	.841

Specifically, we find for one fifth of queries the micro-aspect-based scheme outperforms the session-based scheme. For other queries, they generate very similar rankings. We examine those one fifth of queries and find the situations conform to our analysis in Section 6. Specific investigation results for two queries are shown in Tables 4 and 5. Candidates are anonymized. For the two queries, the micro-aspect-based scheme ranks c_1 and c_3 at the top while the session-based scheme ranks c_2 and c_4 at the top, respectively. In Surf-Lab, we found c_2 was reading

TABLE 3
Comparison of three advisor search schemes on the Surf-IBM dataset. The session-based scheme is intrinsically applying the traditional language model based expert search method on Web surfing data directly.

Method	NDCG@1	NDCG@2	NDCG@3
Micro-aspect-based	**.900**	**.926**	**.928**
Session-based	.817	.885	.894
Task-based	.633	.666	.669

$<$ Thinking in Java $>$ and generated 45 sessions, none of which was focused on the "multithreading" aspect. c_1 generated 3 true relevant sessions and should be regarded as a better candidate. Similar situation is observed for c_3 and c_4 with respect to the query "Mobile Location Network Protocols".

TABLE 4
An example query from Surf-Lab where the micro-aspect-based scheme shows better performance. "True rel / Generally rel" means the number of sessions which are indeed relevant to the query divided by the number of sessions which are relevant to the general domain "Java".

Q: Java multithreading	
Candidate	True rel / Generally rel
c_1	3 / 4
c_2	0 / 45

TABLE 5
An example query from Surf-IBM where the micro-aspect-based scheme shows better performance. "True rel / Generally rel" means the number of sessions which are indeed relevant to the query divided by the number of sessions which are relevant to the general domain "Network Protocols".

Q: Mobile Location Network Protocols	
Candidate	True rel / Generally rel
c_3	8 / 13
c_4	1 / 66

7.3 Session Clustering

Instead of providing a comprehensive comparison of different clustering methods, which is not the focus of this work, the following experiment shows that the proposed GDP model with LE preprocessing (named LEGDP) achieves its design goal. We implement two baseline methods for comparison: (1) Spectral Clustering (SC), which is a popular clustering method requiring users to specify the number of clusters; (2) GDP with Principle Component Analysis (PCAGDP), which first uses PCA to reduce data dimensionality

and then applies GDP on the subspace. We use Surf-Lab in this experiment.

The ground truth task labels were obtained manually. Each student involved in Surf-Lab was given his/her session data and was asked to assign a task label to each session. Three popular evaluation metrics for clustering are employed: *Purity*, *F-measure* and *Normalized Mutual Information* (NMI). Purity tries to map each cluster to the class in the ground truth which is the most frequent in the cluster. It is defined as the accuracy of this map as follows:

$$Purity(\Omega, \Xi) = \frac{1}{n} \sum_{k=1}^{K} \max_{j \in \{1,\ldots,J\}} |\omega_k \cap \xi_j|, \qquad (9)$$

where $\Omega = \{\omega_1, \ldots, \omega_K\}$ is the set of clusters and $\Xi = \{\xi_1, \ldots, \xi_J\}$ is the set of ground truth classes. F-measure operates on pairs of objects (i.e. sessions). The clustering result is treated as $n(n-1)/2$ decisions, each of which corresponds to a pair of objects and decides whether they belong to the same class. Based on this, the precision and recall scores are computed and the F-measure is computed as the harmonic mean of precision and recall. NMI is defined as

$$NMI(\Omega, \Xi) = \frac{I(\Omega, \Xi)}{\frac{1}{2}(H(\Omega) + H(\Xi))}, \qquad (10)$$

where $I(\cdot)$ and $H(\cdot)$ represent mutual information and entropy, respectively.

TABLE 6
Comparison of the numbers of tasks discovered by LEGDP and PCAGDP.

Method	Dataset		
	3p-Lab	5p-Lab	Surf-Lab
Ground Truth	33	47	61
LEGDP	37	45	59
PCAGDP	19	38	44

7.3.1 Performance Comparison

First, we show that LEGDP can roughly capture the number of tasks. To this end, we derive two small datasets from Surf-Lab: (1) 3p-Lab, which contains 3 people and 169 sessions; (2) 5p-Lab, which comprises 5 people and 411 sessions. We tune model parameters on 3p-Lab (details in Section 7.3.2) and continue to use the same set of parameters on 5p-Lab and Surf-Lab. The results are shown in Table 6. As can be seen, LEGDP approximately captures the number of tasks. We also show the number of clusters learned by PCAGDP. PCAGDP cannot well capture the number of tasks. The reason could be that PCA is a linear dimensionality reduction method and may not capture the complex topical variations of a task.

Second, we investigate the clustering performance measured by Purity, F1 and NMI. Table 7 shows the results. For LEGDP and SC, we use the same session affinity matrix $\mathbf{W}$ [27]. Therefore, the intrinsic difference between LEGDP and SC is due to GDP and k-means. We set the parameter k for SC to the number of clusters learned by LEGDP for the sake of fairness. We can see from Table 7 that both LEGDP and SC outperforms PCAGDP. This indicates that the nonlinear manifold structures do exist in tasks. PCA can only capture linear manifold structures and therefore does not perform well. LEGDP and SC can achieve comparable performance, but SC needs the user to pre-specify the parameter k.

7.3.2 Parameter Selection for LEGDP

The parameters of LEGDP include the number of selected eigenvectors in LE (i.e. the dimensionality D), the truncation level K of the variational distribution for the infinite Gaussian mixtures (see appendix) and hyperparameters a_0 and b_0 of GDP which encode the prior knowledge for the variance (variance=1/precision) of each Gaussian component. In this section we explore how to set these parameters in practice.

We vary these parameters and investigate their impact on the number of learned clusters and the clustering performance (F1 and NMI). The results are shown in Figures 5 and 6. For the sake of clarity, we omit the results for 5p-Lab. The trends for 5p-Lab are very similar to those shown in the figures. Figures 5(a) and 6(a) give the results for D. As can be seen, too small D indicates a subspace of low discriminative power and therefore the number of clusters is too small and the performance is bad. On the contrary, too large D leads to the sparsity problem (i.e. curse of dimensionality). Hence, the algorithm learns too many clusters and the performance also starts to decrease. We find $D \in [25, 35]$ generally results in good performance. a_0 and b_0 together give a prior for the cluster precision λ_k, i.e. $E(\lambda_{kj}) = a_0/b_0$. Here we fix $b_0 = 1$ and vary a_0/b_0 by taking the average of the sample precisions of all dimensions as the unit. We find there is a wide range of safe values, i.e. a_0/b_0 between $80*$ and $150*$ (average sample precision), in which LEGDP can achieve reasonable numbers of clusters (Figure 5(b)) and good performance (Figure 6(b)). This implies that the model behavior is not very sensitive to prior hyperparameters. Finally, as shown in Figures 5(c) and 6(c), the parameter T shows little influence on the modeling results, as long as it is assigned a value significantly higher than the real number of tasks.

7.4 Fine-grained Knowledge Mining

In this section, we show the results of fine-grained knowledge mining by the d-iHMM model. We only show the results for Surf-Lab. The observations on Surf-IBM are very similar. The basic iHMM model is employed for comparison. Since it is difficult and

TABLE 7
Performance comparison of different clustering methods on the Surf-Lab dataset.

Method	3p-Lab			5p-Lab			Surf-Lab		
	Purity	F1	NMI	Purity	F1	NMI	Purity	F1	NMI
LEGDP	.768	.956	.772	.751	.953	.764	.667	.947	.698
SC	.762	.948	.753	.721	.948	.733	.658	.933	.675
PCAGDP	.414	.562	.448	.457	.667	.484	.448	.685	.467

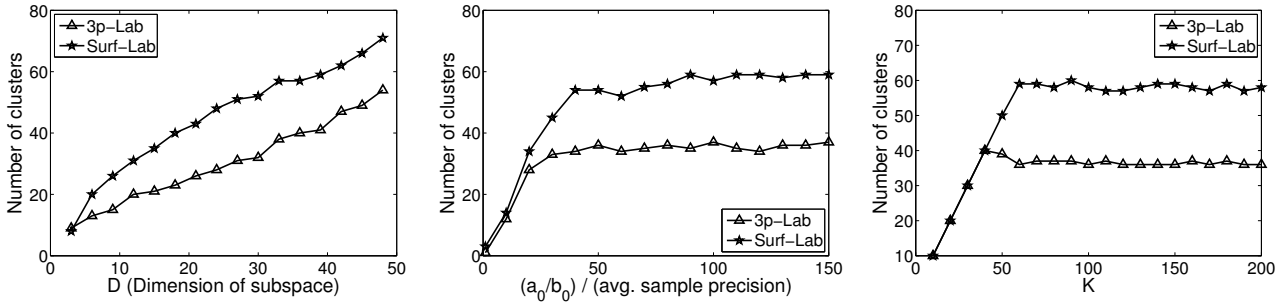


Fig. 5. Number of learned clusters by LEGDP when varying (a) D, (b) a_0 / b_0 and (c) K.

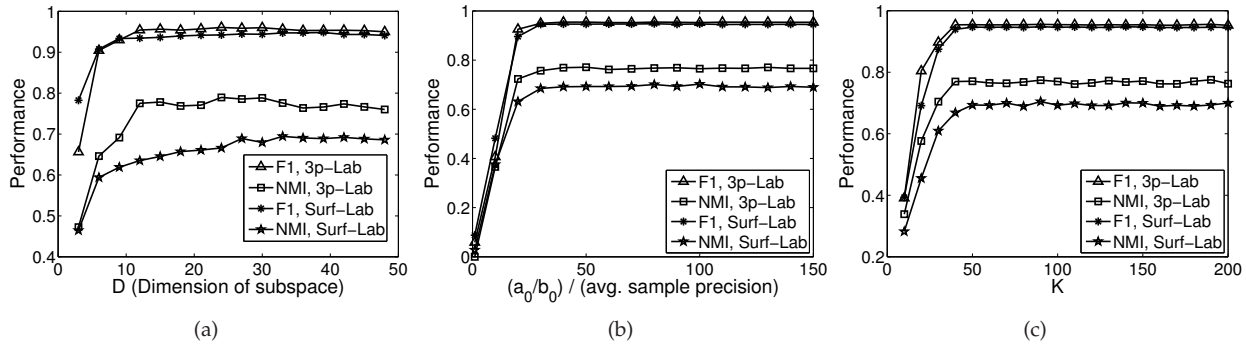


Fig. 6. Performance of LEGDP when varying (a) D, (b) a_0 / b_0 and (c) K.

laborious to perform quantitative evaluation by manual labeling, the models are evaluated by an intuitive measure and case studies. The hyperparameters of d-iHMM and iHMM are set as follows: $h_0 = 0.5$, $a_\alpha = 4$, $b_\alpha = 2$, $a_{\gamma_0} = 3$ and $b_{\gamma_0} = 6$. The additional hyperparameter δ_0 of d-iHMM is set to 0.25.

Our goal is to learn different states (micro-aspects) in a textually cohesive task. The intuition is that the learned states should have low intra-entropies and high inter-distances between them. Therefore, we use the average state entropy divided by the average pairwise distance between states as an intuitive measure to evaluate the results:

$$\frac{\frac{1}{S}\sum_p(-\sum_i p_i \log(p_i))}{\frac{2}{S(S-1)}\sum_p\sum_{q\neq p}\frac{1}{2}(KL(p,q)+KL(q,p))}, \quad (11)$$

where S is the total number of states, $KL(\cdot)$ represents the KL-Divergence function and p, q represent the empirical unigram distributions for two learned states estimated by aggregating all the associated sessions. The unigram distributions are defined on the 100 most frequent words in the task. A good set of states should have a low value of this measure. The results are shown in Figure 7. For some small-size tasks, either d-iHMM or iHMM outputs only one state. The results of these tasks are omitted since the measure is undefined. We can see from Figure 7 that d-iHMM can learn better states for almost all the tasks.

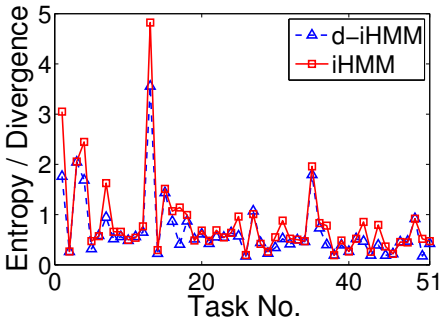


Fig. 7. Comparison of d-iHMM and iHMM with respect to the average state entropy divided by the average pairwise KL-Divergence between states.

The top five words of each learned state's unigram

TABLE 8
Micro-aspects mined by d-iHMM for the task "clustering study."

	Top five words of the unigram model
Background	clustering, endnote, pdf, uc, data
state 1	spectral, clustering, tutorial, algorithms, matrix
state 2	sequence, protein, sequences, oxford, press
state 3	mean, nonparametric, shift, vision, blurring
state 4	detection, evolving, online, news, topic
state 5	pass, single, clustering, new, cluster
state 6	segment, audio, segments, broadcast, segmentation
state 7	algorithm, spatial, density, noise, fuzzy
state 8	smooth, time, gene, function, expression

TABLE 9
Micro-aspects mined by iHMM for the task "clustering study."

	Top five words of the unigram model
state 1	clustering, spectral, pdf, endnote, tutorial
state 2	clustering, algorithm, fuzzy, uc, evolving
state 3	clustering, segment, endnote, line, google
state 4	clustering, segment, news, endnote, music
state 5	clustering, nonparametric, mean, endnote, shift
state 6	endnote, clustering, pdf, algorithm, evolving
state 7	clustering, endnote, single, pass, sequence
state 8	clustering, cluster, pass, single, pdf
state 9	clustering, pdf, endnote, spectral, graph
state 10	clustering, smooth, endnote, uc, time

model for the task "clustering study" are shown in Tables 8 (d-iHMM) and 9 (iHMM). Table 8 shows that d-iHMM successfully summarizes different pieces of fine-grained knowledge into different states. These include different clustering methods such as spectral clustering (state 1), single-pass clustering (state 5) and density based methods (state 7), and different application settings, e.g. protein sequence clustering (state 2), news event detection (state 4). The iHMM model generates 10 states for "clustering study" (Table 9). We find iHMM tends to mess up different micro-aspects. For example, both state 3 and state 4 contain "segment clustering", and both state 7 and state 8 have "single-pass clustering". It sometimes mixes different micro-aspects, e.g. state 7 also contains "sequence clustering". As another example, we show the micro-aspects learned for the task "Programming contests" by d-iHMM and iHMM in Table 10 and 11, respectively. This task is due to a user who was participating in Topcoder[3] contests. Both models learn four states for this task. We can see from Table 10 that this user was learning some classic algorithms and data structures: sorting, maximum flow and black-red trees. The iHMM model again fails to separate different micro-aspects (Table 11). The explanation is that different micro-aspects under the same task share common background textual characteristics. iHMM tries to model the background content in each state independently, which leads to low discriminative power. On the contrary, d-iHMM has higher discriminative power by modeling background words in each state by a common background unigram model.

TABLE 10
Micro-aspects mined by d-iHMM for the task "Programming contests".

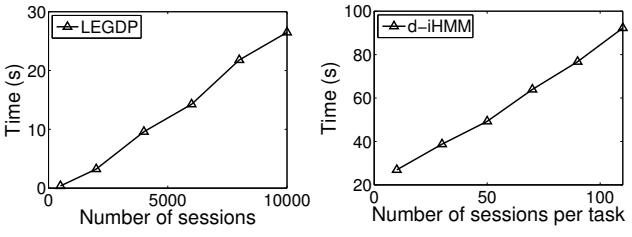
	Top five words of the unigram model
Background	contests, sort, review, black, tree
state 1	black, tree, node, red, nodes
state 2	flow, maximum, search, contests, capacity
state 3	sort, merge, list, insertion, sorting
state 4	class, review, contests, match, track

TABLE 11
Micro-aspects mined by iHMM for the task "Programming contests".

	Top five words of the unigram model
state 1	sort, black, tree, node, data
state 2	flow, contests, review, search, overview
state 3	virtual, contests, review, class, algorithm
state 4	contests, sort, review, black, tree

7.5 Running time

We test the efficiency of the whole framework. We collect more Web surfing packets from IBM's gateway to form datasets of various sizes. The running time of LEGDP and d-iHMM is shown in Figure 8. The tests were run in MATLAB on a PC with Intel Core i7 and 8G memory. The running time of LEGDP seems linear w.r.t the number of sessions, while that of d-iHMM shows a slightly accelerated growth (due to M). The observations conform to the analysis in Section 6. We set the number of sampling iterations to 500 for d-iHMM. d-iHMM is more costly. Fortunately, the computation for different tasks can be parallelized. The advisor search phase only requires a few milliseconds since its main cost is one matrix vector multiplication.



(a) LEGDP (b) d-iHMM

Fig. 8. Running time of LEGDP and d-iHMM.

3. http://www.topcoder.com/

8 Conclusions

We introduced a novel problem, fine-grained knowledge sharing in collaborative environments, which is desirable in practice. We identified digging out fine-grained knowledge reflected by people's interactions with the outside world as the key to solving this problem. We proposed a two-step framework to mine fine-grained knowledge and integrated it with the classic expert search method for finding right advisors. Experiments on real Web surfing data showed encouraging results.

There are open issues for this problem. (1) The fine-grained knowledge could have a hierarchical structure. For example, "Java IO" can contain "File IO" and "Network IO" as sub-knowledge. We could iteratively apply d-iHMM on the learned micro-aspects to derive a hierarchy, but how to search over this hierarchy is not a trivial problem. (2) The basic search model can be refined, e.g. incorporating the time factor since people gradually forget as time flows. (3) Privacy is also an issue. In this work, we demonstrate the feasibility of mining task micro-aspects for solving this knowledge sharing problem. We leave these possible improvements to future work.

Acknowledgments

This research was sponsored in part by the National Natural Science Foundation of China under Grant No. 61373118, the Army Research Laboratory under cooperative agreements W911NF-09-2-0053 and NSF IIS 0954125. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notice herein. The authors would like to thank the editor and reviewers for their constructive comments that helped improve this paper.

References

[1] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, pages 43–50, 2006.
[2] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden markov model. In *Advances in neural information processing systems*, pages 577–584, 2001.
[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.
[4] D. Blei and M. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–143, 2006.
[5] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *NIPS*, 2003.
[6] D. M. Blei and J. D. Lafferty. Dynamic topic models. In *ICML*, pages 113–120, 2006.
[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, 2003.
[8] P. R. Carlile. Working knowledge: how organizations manage what they know. *Human Resource Planning*, 21(4):58–60, 1998.
[9] N. Craswell, A. P. de Vries, and I. Soboroff. Overview of the trec 2005 enterprise track. In *TREC*, 2005.
[10] H. Deng, I. King, and M. R. Lyu. Formal models for expert finding on dblp bibliography data. In *ICDM*, pages 163–172, 2009.
[11] Y. Fang, L. Si, and A. P. Mathur. Discriminative models of integrating document evidence and document-candidate associations for expert search. In *SIGIR*, pages 683–690, 2010.
[12] T. S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.
[13] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.
[14] M. Ji, J. Yan, S. Gu, J. Han, X. He, W. Zhang, and Z. Chen. Learning search tasks in queries and web pages via graph regularization. In *SIGIR*, 2011.
[15] R. Jones and K. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM*, pages 699–708, 2008.
[16] A. Kotov, P. Bennett, R. White, S. Dumais, and J. Teevan. Modeling and analysis of cross-session search tasks. In *SIGIR*, pages 5–14, 2011.
[17] R. Kumar and A. Tomkins. A characterization of online browsing behavior. In *WWW*, pages 561–570, 2010.
[18] J. Liu and N. Belkin. Personalizing information retrieval for multi-session tasks: The roles of task stage and task type. In *SIGIR*, pages 26–33, 2010.
[19] X. Liu, W. B. Croft, and M. Koll. Finding experts in community-based question-answering services. In *CIKM*, pages 315–316, 2005.
[20] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. In *Proceedings of the 38th international conference on very large databases*, pages 716–727, 2012.
[21] M. Medvedovic and S. Sivaganesan. Bayesian infinite mixture model based clustering of gene expression profiles. *Bioinformatics*, 18(9):1194–1206, 2002.
[22] R. M. Neal. Slice sampling. *Annals of statistics*, pages 705–741, 2003.
[23] C. Rasmussen. The infinite gaussian mixture model. In *NIPS*, page 554C560, 2000.
[24] P. Serdyukov, H. Rode, and D. Hiemstra. Modeling multi-step relevance propagation for expert finding. In *CIKM*, pages 1133–1142, 2008.
[25] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.
[26] J. Van Gael, Y. Saatci, Y. Teh, and Z. Ghahramani. Beam sampling for the infinite hidden markov model. In *ICML*, pages 1088–1095, 2008.
[27] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.
[28] H. Wang, Y. Song, M.-W. Chang, X. He, R. White, and W. Chu. Learning to extract cross-session search tasks. In *WWW*, pages 1353–1364, 2013.
[29] R. White, P. Bailey, and L. Chen. Predicting user interests from contextual information. In *SIGIR*, pages 363–370, 2009.
[30] Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.

Ziyu Guan received the BS and PhD degrees in Computer Science from Zhejiang University, China, in 2004 and 2010, respectively. He had worked as a research scientist in the University of California at Santa Barbara from 2010 to 2012. He is currently a full professor in the College of Information and Technology of China's Northwest University. His research interests include attributed graph mining and search, machine learning, expertise modeling and retrieval, and recommender systems.

Shengqi Yang is a PhD candidate at the Computer Science Department, UC Santa Barbara under the supervision of Prof. Xifeng Yan. Prior to joining UCSB, he earned his BS and MS degrees from Beijing University of Posts and Telecoms in 2007 and 2010, respectively. His research interests lie in a broad area of Data Management, with emphasis on managing and querying over large-scale heterogeneous graphs or unstructured data.

Huan Sun received the BS degree in Electronic Engineering and Information Science from the University of Science and Technology of China, in 2010. She is currently working toward the PhD degree in computer science at the University of California, Santa Barbara. Her research interests include statistical machine learning, deep learning, and data mining.

Mudhakar Srivatsa is a research staff member in Network Technologies Department at Thomas J. Watson Research Center. He received his PhD in computer science from Georgia Tech. His research interests primarily include network analytics and secure information flow. He serves a principal investigator for Information Network Research in Network Science Collaborative Technology Alliance where he is working on data mining techniques for co-evolving networks (social, information, and communication).

Xifeng Yan is an associate professor at the University of California at Santa Barbara. He holds the Venkatesh Narayanamurti Chair in Computer Science. He received his Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign in 2006. He was a research staff member at the IBM T. J. Watson Research Center between 2006 and 2008. He has been working on modeling, managing, and mining graphs in bioinformatics, social networks, information networks, and computer systems. His works were extensively referenced, with over 7,000 citations per Google Scholar. He received NSF CAREER Award, IBM Invention Achievement Award, ACM-SIGMOD Dissertation Runner-Up Award, and IEEE ICDM 10-year Highest Impact Paper Award.