Analyzing Expert Behaviors in Collaborative Networks

Huan Sun[1] Mudhakar Srivatsa[2] Shulong Tan[1] Yang Li[1] Lance M. Kaplan[3] Shu Tao[2]
Xifeng Yan[1]
{huansun, shulongtan, yangli, xyan}@cs.ucsb.edu
{msrivats, shutao}@us.ibm.com
lance.m.kaplan.civ@mail.mil
[1]University of California, Santa Barbara, [2]IBM T.J. Watson Research Center, [3]U.S. Army Research Lab

ABSTRACT

Collaborative networks are composed of experts who cooperate with each other to complete specific tasks, such as resolving problems reported by customers. A task is posted and subsequently routed in the network from an expert to another until being resolved. When an expert cannot solve a task, his routing decision (*i.e.*, where to transfer a task) is critical since it can significantly affect the completion time of a task. In this work, we attempt to deduce the cognitive process of task routing, and model the decision making of experts as a generative process where a routing decision is made based on mixed routing patterns.

In particular, we observe an interesting phenomenon that an expert tends to transfer a task to someone whose knowledge is neither too similar to nor too different from his own. Based on this observation, an *expertise difference* based routing pattern is developed. We formalize multiple routing patterns by taking into account both rational and random analysis of tasks and present a generative model to combine them. For a held-out set of tasks, our model not only explains their real routing sequences very well, but also accurately predicts their completion time. Under three different quality measures, our method significantly outperforms all the alternatives with more than 75% accuracy gain. In practice, with the help of our model, hypotheses on how to improve a collaborative network can be tested quickly and reliably, thereby significantly easing performance improvement of collaborative networks.

Source Code
`http://www.cs.ucsb.edu/~huansun/collaborativenet`

1. INTRODUCTION

Collaborative networks are abundant in real life, where experts collaborate with each other to complete specific tasks. In service businesses, a service provider often maintains an expert network where service agents collaboratively solve problems reported by customers. Bugzilla[1] is a bug tracking system where software developers jointly fix the reported bugs in projects. In a classic collaborative network, upon receiving a task, an expert first tries to solve it; if he fails, the expert will route the task to another expert. The task is completed until it reaches an expert who can provide a solution.

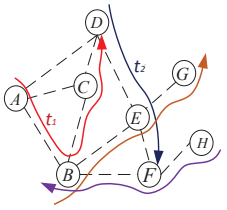

Figure 1: A Sample Collaborative Network

Figure 1 shows a sample collaborative network with task routing examples. Task t_1 starts at expert A and is resolved by expert D, and task t_2 starts at expert D and is resolved by expert F. The sequences $A \rightarrow B \rightarrow C \rightarrow D$ and $D \rightarrow E \rightarrow F$ are called *routing sequences* of task t_1 and task t_2 respectively. The number of experts on a routing sequence measures the completion time of a task and signifies the efficiency of a collaborative network in problem solving. The shorter a routing sequence, the more efficient a network.

When the number of experts in a collaborative network becomes large, to whom an expert routes a task significantly affects the completion time of the task. For example, in Figure 1, task t_1 can be directly routed to the resolver D from A. In this case the routing decision made by expert A is critical. Therefore, understanding how an expert makes a certain routing decision and detecting his routing behavioral patterns will help us identify the inefficiency of a collaborative network.

The task resolution problem in collaborative networks has been studied before. Shao *et al.* [19] propose a sequence mining algorithm to improve the efficiency of task resolution in IT service. Miao *et al.* [13] develop generative models and recommend better routing by considering both task routing sequences and task contents. In [23], Zhang *et al.* study the resolution of prediction tasks, which are to obtain probability assessments for a question of interest. All of these studies aim at developing automated algorithms that can effectively speed up a task's resolution process. However, they largely ignore the human factor in real task routing. Take Figure 1 as an example. Why does expert A route task t_1 to B instead of D? Is it because he does not understand t_1 well, thus randomly distributing it to B, or he believes B has a better chance to solve it, or B has a better chance to find the right expert to solve it? Does expert A make more ratio-

KDD'14, August 24–27, 2014, New York, U.S.A.

nal decisions than random decisions? While it is very hard to infer A's decision logic based on an individual task, it is possible to infer it by analyzing many tasks transferred and solved by A, B and D. In this work, we focus on analyzing real expert networks and try to understand the experts' decision logic, *i.e.*, what kind of routing patterns an expert follows when deciding where to route a task. This understanding will help detect the inefficient spots in a collaborative network and give guidance to the management team to provide targeted expert training.

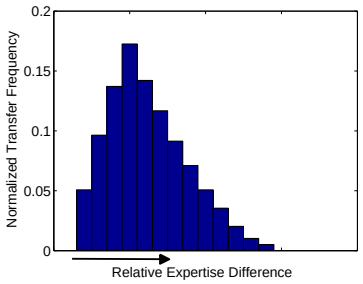

Figure 2: Task Transfer Frequency vs. Expertise Difference.

After analyzing thousands of tasks in an IBM service department, we recognize that in many cases, an expert might not route a task to the *best* candidates (in terms of the possibility to solve the task), especially when the task is far beyond his expertise. Instead, the task is transferred to an expert whose speciality is between the current expert and the best candidates. This routing pattern is clearly indicated by Figure 2. Figure 2 plots the histogram of expertise difference $\frac{\|E_A - E_B\|}{\|E_A\|}$, which is calculated when expert A transfers a task to B. E_A represents the expertise of A and is automatically learnt based on A's task resolution records. It is observed that an expert tends to transfer a task to some expert whose expertise is *neither* too similar to *nor* too different from his own. This phenomenon can be explained as follows: An expert is less likely to transfer a task to another expert whose expertise is very similar, given that the current expert already fails to resolve the task. On the other hand, if the expertise of two experts are very different, they might actually specialize in quite different domains; therefore, an expert might not be clear about the other's speciality and few tasks would be transferred between them. It is like the situation when a computer science professor gets a quantum physics problem, he might consult a CS professor who is working on quantum computing, rather than directly ask a physics professor who is specialized in quantum physics, though the latter might be a better candidate.

Inspired by the above observation, we introduce a routing pattern describing the general trend of expert A transferring a task to B, based on the *expertise difference* between A and B. Apart from this routing pattern, another two are also formalized. For example, when an expert finds there are five candidates to dispatch a task to – all of them can solve the task, who is he going to contact? A straightforward approach is to randomly pick one. An alterative is to look at the capacity of these candidates and route more tasks to an expert who can process more tasks. An expert could follow a certain pattern when deciding where to transfer a task. Different experts might adopt each routing pattern to a different degree and demonstrate different routing behavioral characteristics. This study is going to infer the routing patterns as well as experts' preferences over them, from the historical routing data, and finally give insightful analysis of experts' performance in a collaborative network.

The technical contributions of this work are three-fold: *First*, to the best of our knowledge, we make the first attempt to analyze the routing behaviors of experts in a collaborative network in a large-scale, quantitative manner. We present a general framework to model the decision making and cognitive process of experts, and instantiate the framework with multiple routing patterns potentially followed by an expert. A generative model is then presented to model experts' routing decisions as a result of mixed routing patterns. After trained on a historical task set, the model can uncover experts' underlying decision logic and further explain real routing sequences in a held-out testing set very well.

Second, our analytical model can accurately predict the completion time of a task before actually routing it in the network. On the one hand, we verify that our analysis of experts' routing decision making reflects the real one, in the sense that a task navigated according to our model shall be completed in a similar time as in the real situation. On the other hand, estimating task completion time is important itself, because an accurate estimate of completion time can provide early signals on the "difficulty" or "abnormality" of a task, and managers can allocate more resources and take early actions to deal with such tasks and shorten customer waiting time.

Third, it is usually expensive, if not impossible, to alter real-world collaborative networks for hypothesis testing, *e.g.*, providing more training to a few critical but inefficient experts or changing network structures for better performance. Since our analytical model has shown similar characteristics to the real human routing in collaborative networks, it can be used to conduct virtual hypothesis testing. Through case studies, we discuss how to utilize our model to optimize collaborative networks.

The rest of the paper is organized as follows. In Section 2, we briefly describe our problem setting. In Section 3, we discuss multiple routing patterns potentially adopted by an expert, followed by Section 4, where a generative model combining these patterns together is presented. Section 5 details the estimation of task completion time, which serves as both an evaluation metric of our model and an important problem our model can deal with. Section 6 presents our detailed experimental results. Related work is reviewed in Section 7. We conclude this work in Section 8.

2. PRELIMINARIES

We first clarify the notations used in this work. $\mathcal{E} = \{e_1, e_2, ..., e_i, ..., e_G\}$ is a set of experts in a collaborative network. Neighbors of expert e_i refers to those experts e_i has ever transferred tasks to. $\mathcal{N}_i$ denotes the 1-hop neighborhood of expert e_i. $\mathcal{W} = \{w_1, w_2, ..., w_n, ..., w_N\}$ is a set of words used to describe the tasks. $\mathcal{T} = \{t_1, t_2, .., t_m, ..., t_M\}$ is a set of tasks resolved by the collaborative network, where each t_m is an $N \times 1$ word vector with each dimension recording the word frequency in the task description. Apart from the textual description, each task is also associated with a

routing sequence starting from an initial expert to the resolver of the task.

ID	Entry	Time	Expert
599	New ticket: the available space on the /var file system is low	9/14/06 5:57:16	IN039
599	...(operations by IN039)...	...	IN039
599	Ticket 599 transferred to SAV59	...	IN039
599	...(operations by SAV59)...	...	SAV59
599	Ticket 599 transferred to SAV4F	...	SAV59
599	...(operations by SAV4F)...	...	SAV4F
599	Problem resolved: free up disk space in the file system	9/14/06 9:57:31	SAV4F

Table 1: The Lifetime of An Example Task.

Table 1 shows one example problem ticket in an IT service department. The ticket with ID 599 is a problem related to *operating system*, specifically, *the low percentage of the available file system space*. It was assigned to or initiated by expert IN039, then routed through expert SAV59, and finally resolved by expert SAV4F.

In this paper, we study the following problems: *How does an expert in a collaborative network make a routing decision? Is there any pattern an expert generally follows when routing a task?* Different from previous studies [13, 19, 23], we do not propose algorithms to perform more efficient routing. Instead, we hope to understand the routing decisions *actually* made by the experts in a collaborative network.

3. MODELING ROUTING DECISIONS

We advocate a general and extensible methodology to analyze the routing decision making process, which consists of two types of routing strategies: *Task-Neutral Routing* (TNR) and *Task-Specific Routing* (TSR). Under the Task-Neural Routing, an expert does not take into account the specificity of a task when making a routing decision, and treat different tasks equivalently. Under the Task-Specific Routing, however, an expert makes a routing decision by analyzing the specific task being transferred.

We attempt to deduce the cognitive process of an expert during task routing, and model the decision making of an expert as a generative process where a routing decision is generated based on the two types of routing strategies. Each routing type is instantiated with three basic routing patterns. Our generative model, with multiple routing patterns as mixture components, is then proposed to describe the process of an expert's decision making.

3.1 Routing Patterns

Given a task, an expert e_i first establishes a pool of candidates, C, to dispatch the task to. The difference between the Task-Neutral Routing (TNR) and the Task-Specific Routing (TSR) lies in the composition of C. In terms of TNR, C contains all the experts e_i has contacted before, i.e., all of his neighbors $\mathcal{N}_i$. In terms of TSR, C is limited to experts who are able to solve the current task in $\mathcal{N}_i$. TNR accommodates the situations when an expert does not understand a task very well, or an expert has a careless work attitude, and thereby making a routing decision irrespective of the specific task and its possible resolvers; whereas TSR mimics the situation where an expert assesses others' ability to solve a task before dispatching it.

Once C is established, e_i selects one expert from C to route the task to, based on a certain routing pattern. We identified three basic routing patterns:

Uniform Random (UR). This routing pattern implies that an expert makes a decision by randomly selecting one of the candidates in C with an equal probability.

$$P(e_i \xrightarrow{t} e_j \mid \text{UR}) = \mathbb{1}(e_j \in C)\frac{1}{|C|} \tag{1}$$

Where $e_i \xrightarrow{t} e_j$ denotes the event that expert e_i transfers task t to e_j. $\mathbb{1}$ is an indicator function: $\mathbb{1}(e_j \in C)$ picks value 1 if $e_j \in C$ holds otherwise 0;

Volume-biased Random (VR). Under this routing pattern, an expert also randomly dispatches tasks, but with a rate proportional to the volume of tasks previously dispatched. VR mimics the situation where the task processing capacity could vary for different experts. VR can be regarded as an expert's reaction when he finds tasks are processed slowly by some of his collaborators. Let volume v_{ij} denote the number of transferred tasks from expert e_i to e_j.

$$P(e_i \xrightarrow{t} e_j \mid \text{VR}) = \mathbb{1}(e_j \in C)\frac{v_{ij}}{\sum_{e_k \in C} v_{ik}} \tag{2}$$

Expertise Difference (EX). In addition to the above two routing patterns, this one is derived from the observation shown in Figure 2: An expert is more likely to send a task to another expert whose expertise is neither too close nor too far from his own. Let f_{ij} denote the general trend of expert e_i sending a task to e_j, given their expertise.

$$P(e_i \xrightarrow{t} e_j \mid \text{EX}) = \mathbb{1}(e_j \in C)\frac{f_{ij}}{\sum_{e_k \in C} f_{ik}} \tag{3}$$

To estimate f_{ij}, we build a model based on the observation in Figure 2. The relative expertise difference between expert e_i and e_j, is calculated as $\Delta(e_i, e_j) = \frac{\|e_i - e_j\|}{\|e_i\|}$, where for simplicity e_i is reused to represent the expertise vector of expert e_i. The expertise estimation problem will be discussed in Section 3.3. Based on Figure 2, we assume for those tasks transferred, the relative expertise difference between a task sender and a task receiver follows a log-normal distribution with parameters μ and σ^2. We made this assumption due to the non-negative nature of $\Delta(e_i, e_j)$ and the asymmetric shape of the distribution. However, in our experiments, we also test a model with a normal distribution to estimate f_{ij} and show that the log-normal distribution works better. Under the log-normal distribution, expert e_i is more likely to transfer tasks to e_j once $\Delta(e_i, e_j)$ obtains a higher probability density; therefore, f_{ij} is estimated by:

$$f_{ij} \propto \frac{1}{\Delta(e_i, e_j)} e^{-\frac{[\ln \Delta(e_i, e_j) - \mu]^2}{2\sigma^2}}. \tag{4}$$

Note that the histogram in Figure 2 is the accumulation of all possible routing patterns, not only the *expertise difference* pattern. Later, in the experiments, we will show that the EX pattern with μ and σ^2 directly estimated from

Figure 2 does not perform the best. Instead, parameters μ and σ^2 shall be estimated particularly based on tasks transferred following the EX pattern, which are not known a priori. We will learn them simultaneously through a mixture model with all three routing patterns considered.

3.2 Task-Specific Routing

TNR does not take into account the distinctiveness of a specific task. In contrast, TSR means that an expert makes a routing decision based on the specific task and matches it with potential resolvers. Given expert e_i to transfer task t, his routing decision under TSR is influenced by two factors: (1) whether an expert can solve the task or not, and (2) how "familiar" e_i is with that expert. The first factor, only related to the task itself, can be conducted by a classification process without involving e_i. The classification identifies a subset of e_i's neighbors who can solve the task, as the candidate pool C. We build the classifier based on the task resolution records of experts in Section 3.3. The second factor is a human factor which can be modeled by the same routing patterns we previously introduced such as UR, VR, and EX. Overall, TSR will check the neighborhood of e_i, run the classifier, and establish a set of candidates C who are capable of solving t. It then selects one particular candidate from C based on one of {UR, VR, EX}. In a special case where $C = \emptyset$, TSR is reduced to TNR where we simply use the entire neighborhood set $\mathcal{N}_i$ as the candidate pool, *i.e.*, $C = \mathcal{N}_i$.

To summarize, the three basic routing patterns {UR, VR, EX} combined with the Task-Neutral Routing type and Task-Specific Routing type, generate six particular routing patterns: TNR^{ur}, TNR^{vr}, TNR^{ex}, TSR^{ur}, TSR^{vr}, and TSR^{ex}.

The success of our work is related to the recognition of human factors in task routing. A straightforward Task-Specific Routing strategy is to transfer tasks to an expert with the highest probability to solve it. Our experiments show that such a routing strategy cannot capture the real characteristics of human decision making, such as randomness, uncertainty, and sub-optimality. For example, we observe that similar tasks are often routed to very different experts. Moreover, one expert usually does not search for people who are most likely to solve a problem due to *e.g.*, unfamiliarity with those people; instead, he might select a close collaborator who might be able to solve the problem, but not necessarily with the highest probability.

3.3 Expertise Estimation

In TSR strategy and the EX routing pattern, we need to estimate an expert's expertise and capability to solve a task. Intuitively, the capability depends on both the expert's expertise and the task description. We resort to a classic logistic regression model [6] that takes an expert's expertise vector and a task's word vector as input, and outputs the expert's capability to solve the task. In the logistic model, the probability for expert e_i to solve task t, denoted as $P(e_i, t)$, is defined as follows:

$$P(e_i, t) = \frac{1}{1 + \exp(-(W_1 t + W_2 e_i + b))} \tag{5}$$

For simplicity, the expertise vector for expert e_i is of the same length as task t, *i.e.*, an $N \times 1$ vector. W_1 and W_2 are the $1 \times N$ weights respectively associated with the word vector of a task and the expertise vector of an expert. Each component of W_1 and W_2 denotes the contribution of the corresponding dimension in t or e_i to the capability prediction. b is a bias scalar in the logistic model. The expertise vectors e_i's are not known a priori and they are to be estimated together with the model parameters $\{W_1, W_2, b\}$. We use $W = \{W_1, W_2, b, e_i's\}$ to denote all the parameters.

Given a task t and its routing sequence, *e.g.*, $e_i \rightarrow ... \rightarrow e_k$, we observe the groundtruth regarding the resolution capability: The last expert e_k solves t and any other expert on the sequence does not solve it. Therefore, we can formulate a training dataset composed of <expert, task> pairs as instances and $\{0, 1\}$ as the observed probability of an expert to solve a task, *e.g.*, 0 for <e_i, t> while 1 for <e_k, t>.

The optimal solution of parameters in the model is obtained by minimizing the cross-entropy error function [6], based on the <expert, task> pairs in the training dataset :

$$\begin{aligned} \arg\min_{W} \sum_{<e_i,t>} & [-P^*(e_i, t) \log P(e_i, t) \\ & - (1 - P^*(e_i, t)) \log(1 - P(e_i, t))] \end{aligned} \tag{6}$$

where $P^*(e_i, t)$ is the observed probability for expert e_i to solve task t. After the parameters are learnt, given a task and an expert, one can predict the probability for the expert to solve the task using Eqn. 5. Under Task-Specific Routing, when transferring task t, expert e_i identifies a subset of his neighbors $\mathcal{N}_i$ as the candidate pool $C = \{e_j \in \mathcal{N}_i : P(e_j, t) \geq \delta\}$, where δ is set at 0.5 in our implementation. Experts in C have a probability to solve t larger than a threshold, and are estimated capable to solve the task.

Essentially, the expert capability estimation is casted as a traditional classification problem. The logistic model we employed is very similar to classification using SVM [8] with a linear kernel [6]. The difference lies in that the expertise vector of an expert is not known a priori. One might consider using the average word vector of the tasks resolved by an expert to represent the expert's knowledge. However, in practice, this method can be problematic, because we observe that there might be many experts in a network that serve as "intermediate transferrers" and did not resolve any tasks. In our case, we can also utilize those tasks unresolved by an expert to estimate his expertise in Eqn. 6. Besides, by optimizing the cost function over the expertise vectors, we can give lower cost function than fixing them at somewhere non-optimal.

4. GENERATIVE MODEL

In this section, we present a generative model to put the previously discussed routing patterns together and describe an integrated decision making process.

Figure 3 shows the graphical representation of our generative model. We first clarify the notations in the figure as follows: (1) $|\mathcal{E}|$ denotes the number of experts while $|\mathcal{T}_i|$ is the number of tasks expert $e_i \in \mathcal{E}$ has ever transferred. A plate means replicating a process for multiple times. (2) θ_i is the $K \times 1$ mixture weights of different routing patterns for expert e_i, where K is the number of routing patterns. In our current setting, we have $K = 6$ routing patterns, from TNR^{ur} to TSR^{ex}. The k-th component of θ_i reveals the probability that the k-th routing pattern is adopted by e_i to transfer a task. (3) α, a $K \times 1$ vector, is parameters in a Dirichlet prior, and serves as a constraint of all the mixture weights θ_i's. A Dirichlet prior for the mixture weights

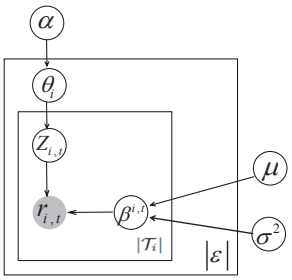

Figure 3: Graphical Representation of Our Model.

tends to alleviate over-fitting problems [7]. Besides, with the Dirichlet prior, the mixture weight θ_{new} for a new expert can be naturally assigned. (4) $Z_{i,t}$ is the label of the routing pattern employed by expert e_i when transferring task t. (5) $\beta^{i,t}$, a $K \times |\mathcal{N}_i|$ matrix, defines the probability distribution of expert e_i transferring task t to an expert in his neighborhood $\mathcal{N}_i$, under K routing patterns. Particularly, each row of $\beta^{i,t}$ is filled by a probability distribution under one of the six routing patterns, as defined in Section 3.1. For experts in $\mathcal{N}_i$ but not in the candidate pool C, the corresponding elements in $\beta^{i,t}$ are naturally filled with 0. For patterns irrelevant to EX, we pre-compute their probability distributions and fill corresponding rows of $\beta^{i,t}$, while TNR^{ex} and TSR^{ex} are parameterized with μ and σ^2. Note that $\beta^{i,t}$ is in the inner plate of the graphical model because $\beta^{i,t}$ is associated with expert e_i and task t. (6)The shaded variable $r_{i,t}$ indicates the observed receiver of task t transferred from expert e_i.

Figure 3 conveys that expert e_i decides where to route task t based on multiple routing patterns $\beta^{i,t}$ and his preference θ_i towards adopting different routing patterns. Now we formally describe the generative process as follows:

For each expert e_i to transfer tasks,

- Draw the mixture weights of K routing patterns: $\theta_i \sim \mathbf{Dir}(\alpha)$.
- For each task t to be transferred by expert e_i,
  - Draw a pattern label: $Z_{i,t} \sim \mathbf{Mult}(\theta_i)$.
  - Draw an expert from $\mathcal{N}_i$ to receive t: $r_{i,t} \sim P(e_i \xrightarrow{t} e_j | Z_{i,t}, \beta^{i,t}), \forall e_j \in \mathcal{N}_i$.

For each task $t \in \mathcal{T}_i$, the transfer relationship for t is represented by $e_i \xrightarrow{t} r_{i,t}$. We formulate the likelihood of observing all the task transfer relationships as follows:

$$\mathcal{L} = P(e_i \xrightarrow{t} r_{i,t}, \forall t \in \mathcal{T}_i, \forall e_i \in \mathcal{E} | \alpha, \mu, \sigma^2) \tag{7}$$

Since a routing decision of an expert is independent from that of another expert while the routing decisions of the same expert for different tasks are not independent from each other, we can rewrite $\mathcal{L}$ in the following way:

$$\begin{aligned} \mathcal{L} &= \prod_{e_i \in \mathcal{E}} P(e_i \xrightarrow{t} r_{i,t}, \forall t \in \mathcal{T}_i | \alpha, \mu, \sigma^2) \\ &= \prod_{e_i \in \mathcal{E}} \int_{\theta_i} P(\theta_i | \alpha) P(e_i \xrightarrow{t} r_{i,t}\ ,\ \forall t \in \mathcal{T}_i | \theta_i, \mu, \sigma^2)\ d\theta_i \end{aligned} \tag{8}$$

Here,

$$\begin{aligned} &P(e_i \xrightarrow{t} r_{i,t}\ ,\ \forall t \in \mathcal{T}_i \mid \theta_i, \mu, \sigma^2) \\ &= \prod_{t \in \mathcal{T}_i} \{ \sum_{Z_{i,t}} P(Z_{i,t} | \theta_i) P(e_i \xrightarrow{t} r_{i,t} \mid Z_{i,t}, \beta^{i,t}) \} \end{aligned} \tag{9}$$

where $P(Z_{i,t}|\theta_i) = \theta_{i,k}$ and $P(e_i \xrightarrow{t} r_{i,t} \mid Z_{i,t}, \beta^{i,t}) = \beta^{i,t}_{k,r}$, if $Z_{i,t} = k$, *i.e.*, the k-th routing pattern is adopted. $\beta^{i,t}_{k,r}$ is the probability for e_i routing task t to expert $r_{i,t}$, under the k-th pattern. $\beta^{i,t}_{k,r}$ shall contain parameters μ and σ^2 if the k-th pattern is TNR^{ex} or TSR^{ex}.

Finally, we resort to the maximum likelihood estimation approach to optimize the parameters in the model:

$$\arg \max_{\alpha, \mu, \sigma^2} \log \mathcal{L} \tag{10}$$

4.1 Inference

Now we discuss how to estimate the model parameters that best fit the observed data. The latent variables are not independent of each other, which makes their true posterior distributions computationally intractable. In this section, we employ a variational approach [6] to solve our model.

4.1.1 Variational Inference

We introduce a variational distribution Q in which the latent variables are independent of each other to approximate their true posterior distribution, *i.e.*, $Q(\theta, Z) = Q(\theta)Q(Z)$, where $\theta = \{\theta_i, \forall e_i \in \mathcal{E}\}$ and $Z = \{Z_{i,t}, \forall e_i \in \mathcal{E}, t \in \mathcal{T}_i\}$. According to the variational distribution, $Q(\theta_i) \sim \mathbf{Dir}(\gamma_i)$, $Q(Z_{i,t}) \sim \mathbf{Mult}(\phi^{i,t})$, , where γ_i and $\phi^{i,t}$ are $K \times 1$ variational parameters. γ_i and $\phi^{i,t}$ have significant meanings where γ_i represents the variational prior for θ_i and reflects which routing pattern e_i tends to adopt, while $\phi^{i,t}$ is the variational posterior mixture weights of different routing patterns adopted by e_i when transferring task t. Given the observed data, both γ_i and $\phi^{i,t}$ will be derived automatically.

Under the variational distribution and Jensen's inequality, we can maximize the lower bound of the log likelihood, instead of directly maximizing $\mathcal{L}$ which is intractable.

$$\log \mathcal{L} \geq E_Q \log P(\mathcal{D}, \theta, Z | \alpha, \mu, \sigma^2) + H(Q) = \lfloor \mathcal{L} \rfloor \tag{11}$$

where $\mathcal{D}$ denotes all the observed task transfer relationships. We expand the lower bound of the log likelihood as follows:

$$\begin{aligned} \lfloor \mathcal{L} \rfloor = &\sum_{e_i \in \mathcal{E}} E_Q \log P(\theta_i | \alpha) + \sum_{e_i \in \mathcal{E}} \sum_{t \in \mathcal{T}_i} E_Q \log P(Z_{i,t} | \theta_i) \\ &+ \sum_{e_i \in \mathcal{E}} \sum_{t \in \mathcal{T}_i} E_Q \log P(e_i \xrightarrow{t} r_{i,t} | Z_{i,t}, \mu, \sigma^2) \\ &+ H(Q(\theta, Z)) \end{aligned} \tag{12}$$

Each term on the right-hand side of the above equation, is a function over the model parameters as shown in Eqn. 13 to Eqn. 16.

$$\begin{aligned} &\sum_{e_i \in \mathcal{E}} E_Q \log P(\theta_i | \alpha) \\ &= -|\mathcal{E}| \log B(\alpha) + \sum_{e_i \in \mathcal{E}} \sum_k (\alpha_k - 1)[\psi(\gamma_{i,k}) - \psi(\sum_k \gamma_{i,k})] \end{aligned} \tag{13}$$

where $B(\alpha) = \frac{\prod_k \Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)}$ is the normalization constant of the Dirichlet distribution $\mathbf{Dir}(\alpha)$.

$$\sum_{e_i\in\mathcal{E}}\sum_{t\in\mathcal{T}_i} E_Q \log P(Z_{i,t}|\theta_i) = \sum_{e_i\in\mathcal{E}}\sum_{t\in\mathcal{T}_i}\sum_k \phi_k^{i,t}[\psi(\gamma_{i,k}) - \psi(\sum_k \gamma_{i,k})] \tag{14}$$

The third term

$$\sum_{e_i\in\mathcal{E}}\sum_{t\in\mathcal{T}_i} E_Q \log P(e_i \xrightarrow{t} r_{i,t}|Z_{i,t},\mu,\sigma^2) = \sum_{e_i\in\mathcal{E}}\sum_{t\in\mathcal{T}_i}\sum_k \phi_k^{i,t}\log\beta_{k,r}^{i,t} \tag{15}$$

As discussed in Eqn. 9, $\beta_{k,r}^{i,t}$ contains parameters μ and σ^2 if the k-th pattern is TNR^{ex} or TSR^{ex}.
The entropy term

$$\begin{aligned} &H(Q(\theta,Z)) \\ &= -\sum_{e_i\in\mathcal{E}}[E_Q\log Q(\theta_i|\gamma_i) + \sum_{t\in\mathcal{T}_i} E_Q \log Q(Z^{i,t}|\phi^{i,t})] \\ &= \sum_{e_i\in\mathcal{E}}[\log B(\gamma_i) - \sum_k(\gamma_{i,k}-1)(\psi(\gamma_{i,k}) - \psi(\sum_k\gamma_{i,k}))] \\ &- \sum_{e_i\in\mathcal{E}}\sum_{t\in\mathcal{T}_i}\sum_k \phi_k^{i,t}\log\phi_k^{i,t} \end{aligned} \tag{16}$$

4.1.2 Parameter Estimation

The model parameters are estimated by using the variational expectation-maximization (EM) algorithm. In the E-step, we update the variational parameters $\{\gamma\text{'s}, \phi\text{'s}\}$ while in the M-step, we update the model parameters α, μ, and σ^2 so that $\lfloor\mathcal{L}\rfloor$ is maximized.

Specifically, the E-step updates the variational parameters according to Eqn. 17 and 18.

$$\phi_k^{i,t} \sim \beta_{k,r}^{i,t}\exp(\psi(\gamma_{i,k}) - \psi(\sum_k\gamma_{i,k}) - 1) \tag{17}$$

$$\gamma_{i,k} = \alpha_k + \sum_{t\in\mathcal{T}_i}\phi_k^{i,t} \tag{18}$$

During the M-step, we maximize the lower bound over the parameter α, μ, and σ^2, by utilizing the classic L-BFGS optimization algorithm [12]. The derivatives over the parameter α are calculated in Eqn. 19.

$$\frac{\partial\mathcal{L}}{\partial\alpha_k} = |\mathcal{E}|[-\psi(\alpha_k) + \psi(\sum_k\alpha_k)] + \sum_{e_i\in\mathcal{E}}[\psi(\gamma_{i,k}) - \psi(\sum_k\gamma_{i,k})] \tag{19}$$

Derivatives over μ and σ^2 depend on the routing pattern TNR^{ex} and TSR^{ex}, as well as the mixture weights corresponding to the two patterns.

$$\frac{\partial\mathcal{L}}{\partial\mu} = \sum_{e_i\in\mathcal{E}}\sum_{t\in\mathcal{T}_i}\sum_{k=1}^{2}\mathbb{1}(r_{i,t}\in C_k)\times\frac{\phi_k^{i,t}}{\beta_{k,r}^{i,t}}\times\frac{f_{ir}\sum_{e_j\in C_k}X_{ij}}{\sigma^2(\sum_{e_j\in C_k}f_{ij})^2} \tag{20}$$

$$\frac{\partial\mathcal{L}}{\partial\sigma^2} = \sum_{e_i\in\mathcal{E}}\sum_{t\in\mathcal{T}_i}\sum_{k=1}^{2}\mathbb{1}(r_{i,t}\in C_k)\times\frac{\phi_k^{i,t}}{\beta_{k,r}^{i,t}}\times\frac{f_{ir}\sum_{e_j\in C_k}Y_{ij}}{2(\sigma^2\sum_{e_j\in C_k}f_{ij})^2} \tag{21}$$

where we assume TNR^{ex} and TSR^{ex} are the 1st and 2nd mixture component respectively. C_k is the candidate pool established under TNR or TSR by e_i when routing task t. f_{ir} is the general trend of e_i sending a task to $r_{i,t}$, based on $\Delta(e_i, r_{i,t})$. $X_{ij} \doteq f_{ij}(\ln\Delta(e_i,r_{i,t}) - \ln\Delta(e_i,e_j))$ and $Y_{ij} \doteq f_{ij}[(\ln\Delta(e_i,r_{i,t}) - \mu)^2 - (\ln\Delta(e_i,e_j) - \mu)^2]$.

The E-step and M-step are performed iteratively until the algorithm converges, which indicates that the current model parameters fit the observed training data.

5. TASK COMPLETION TIME

In a real collaborative network, a task is routed and completed as long as it reaches an expert who can solve it. The completion time (CT) of a task is defined as the number of experts in its routing sequence. Estimation of the completion time *before actually routing a task* is critically useful, as it can raise attention for those troublesome tasks and ask the network allocate more resources to handle such tasks. The estimated completion time can also be used to evaluate routing models. A good routing model shall reflect the real decision making process and give the estimation as accurate as possible.

Experts that can resolve a task are not unique and are not known before the task is actually routed. For a new task, one cannot estimate its completion time by targeting a unique "resolver". Instead, we need to consider multiple potential resolvers and multiple routing sequences.

Given a task and its initial expert, our generative model can generate a routing sequence of experts to process the task. Specifically, given a task t and its current holder, e_i, the receiver $r_{i,t}$ can be sampled according to our generative process described in Section 4. Once $r_{i,t}$ is obtained, it is treated as the current holder of t; the same procedure is repeated to produce the next receiver, until we have L experts to process t in sequence. Although the initial expert to deal with a task is important, in our work, we do not particularly deal with the assignment of an initial expert to a certain task. We assume that the initial expert to a task is given beforehand; it is either decided by the task requestor (*e.g.*, a customer) or by the system.

Task t will stop routing once an expert can solve it. Since each expert in the routing sequence has a probability to solve the task, the completion time can be estimated ($\widehat{\text{CT}}_t$) as the expected number of experts having accessed the task when it is solved.

$$\widehat{\text{CT}}_t = \sum_{m=1}^{L} m\prod_{n=1}^{m-1}(1 - P(r_n,t))P(r_m,t) \tag{22}$$

where r_m (r_n) is the m-th (n-th) expert in a routing sequence. $\prod_{n=1}^{m-1}(1-P(r_n,t))P(r_m,t)$ gives the probability for the m-th expert in the sequence to solve the task while the previous $m-1$ experts fail to, where $P(r_m,t)$'s are estimated using Eqn. 5. Since the probability diminishes quite quickly, we set $L = 10$ in practice. Indeed, $\widehat{\text{CT}}_t$ is not going to change much when L is beyond 10.

6. EXPERIMENTS

In this section, we validate the expertise difference routing pattern and evaluate the accuracy of our method in modeling expert behaviors on various real-life datasets. We will further demonstrate that with the help of our model, better recommendations on expert training could be automatically obtained and provided to managers for improving the performance of collaborative networks.

Our method was implemented in MATLAB. All the experiments were conducted on a 3.4GHZ, 16GB, Intel PC running Windows 7.

6.1 Datasets

We use real-world problem ticket data collected from a problem ticketing system in an IBM IT service department

throughout 2006. Three datasets in different problem categories are explored: DB2, WebSphere, and AIX. DB2 contains problem tickets on database usage and management; WebSphere is a set of problem tickets on the enterprise software IBM WebSphere[2]; and AIX is the category of problem tickets on operating systems.

Datasets	# of tasks	# of experts	% of tasks with CT			
			$=2$	$=3$	$=4$	≥ 5
DB2	26,740	55	44.2	34.3	16.5	5.0
WebSphere	65,786	234	39.0	36.2	20.0	4.8
AIX	120,780	404	40.0	39.4	14.2	6.4

Table 2: Three Datasets on Ticket Resolution.

The details of the three datasets, *i.e.*, the number of tasks, experts, and the distribution of completion time (CT), are shown in Table 2. The three datasets involve approximately 50 to 400 experts. Understanding how an expert makes a certain routing decision among many candidates is a meaningful yet potentially challenging problem. As evident in these datasets, the completion time for different tasks possesses large diversity, which drives us to analyze expert routing behaviors that possibly lead to such diversity. For each dataset, we randomly partition it into two disjoint subsets: 75% of tasks for training, 25% for testing.

6.2 Evaluation Measures

To evaluate our generative model in capturing the real decision making process of an expert, we employ two types of measures: (1) *Routing Sequence Likelihood.* We compute the log likelihood (LL) of the routing relationships in the held-out testing dataset, according to Eqn 8. The higher the log likelihood, the better a model explains the routing decisions of experts. (2) *Predicted Completion Time.* The routing decision of an expert significantly affects the completion time of a task. Our model is considered valid if a task routed according to our generative process can achieve a similar completion time as it does in real situations. Two measures are employed to calculate the difference between the estimated, $\widehat{\text{CT}}$ and the real completion time, CT.

1. Mean Absolute Error (MAE).

$$\text{MAE} = \frac{1}{|\text{Test Set}|} \sum_{t \in \text{Test Set}} |\widehat{\text{CT}}_t - \text{CT}_t|. \quad (23)$$

2. Step Loss Measure (SL). Instead of directly computing the difference between $\widehat{\text{CT}}_t$ and CT_t as errors, step loss measure [16] incorporates some tolerance of the difference. If the difference is larger than the tolerance, one estimation mistake is made. We set the tolerance in our case as 1. That is, if the difference between $\widehat{\text{CT}}_t$ and CT_t is within 1, the estimation is regarded as correct; otherwise, it is regarded as wrong. We calculate the percentage of the wrong estimations in the testing data set. The lower, the better.

$$\text{SL} = \frac{1}{|\text{Test Set}|} \sum_{t \in \text{Test Set}} \mathbb{1}(|\widehat{\text{CT}}_t - \text{CT}_t| > 1) \quad (24)$$

where $\mathbb{1}$ is an indicator function: it picks value 1 if the condition $|\widehat{\text{CT}}_t - \text{CT}_t| > 1$ holds otherwise 0.

6.3 Evaluation Results

We compare our model with the following algorithms.
(1) Regression: For each task, to estimate its completion time, one can resort to a regression algorithm to make the prediction. We use two classic methods: Support Vector Regression (SVR) [8] and Bayesian regression method [15]. Given a task, two types of features are input to each method: (i) word frequency vector in the description of a task; (ii) the initial expert assigned to the task. 10-fold cross validation is conducted for both methods. We evaluate SVR with different kernels including a linear kernel, a polynomial kernel, an RBF kernel and a wavelet kernel. For Bayesian regression, we consider Bayesian linear regression and Bayesian logistic regression. Classification using SVM [8] or naive Bayes classifier [15] are also tested, which turns out to be worse than the regression methods. Among all the variants of SVR or Bayesian regression, we always show their best results obtained. The classification/regression approaches are employed as straightforward methods for completion time estimation. They do not attempt to understand the decision making process of experts, and their results on the sequence likelihood measure are not available.
(2) Generative models. Miao *et al.* [13] estimate the probability of an expert to solve a task and the probability of transferring a task from an expert to another. Given a task, [13] recommends a sequence of experts to route the task. Their goal is to shorten the routing as much as possible, while our goal is to characterize human routing patterns in the real network.

6.3.1 Model Accuracy

Table 3 summarizes the performance of all the methods on step loss measure, MAE, and log likelihood. From the results of SVR and Bayesian regression, we can see that the completion time of a task cannot be accurately predicted based on the straightforward regression methods. In fact, we observe that in the real datasets, similar tasks, even if assigned initially to the same expert, are often routed to different experts and resolved with different completion time. This implies the resolution of a task is a complicated process and involves human factors. The estimated completion time in [13] is usually shorter than the real one as its goal is to shorten routing sequences. It is not surprising that it incurs a large step loss and MAE.

Now we test multiple variants of our generative model. For each variant, we select combinations of different routing patterns to train a generative model, and test the learned model under the three measures.

We first examine the performance of TNR-related and TSR-related routing patterns separately. Then they are combined together as TNR+TSR. Table 3 clearly shows both TNR and TSR play a critical role to reduce SL and MAE, indicating both types of strategies are adopted by experts in real cases. Our model does capture the decision making process of experts in a collaborative network. Our method significantly outperforms the content-only classification methods by **75**%. Moreover, the MAE between our estimated completion time and the real one is between **0.07** and **0.15**, which shows that our method can be used to accurately predict the task completion time.

We then experiment if the expertise difference (EX) routing pattern makes sense. Specifically, we test the model that combines all the routing patterns except TNR^{ex} and TSR^{ex},

denoted as TNR+TSR-EX. The results indicates that with the EX routing pattern considered, TNR+TSR will better capture the real decision making process. This result can be attributed to our observation: an expert is more likely to transfer a task to some expert whose expertise is neither too similar nor too different.

DB2			
Models	Step Loss (%)	MAE	LL($\times 10^4$)
TNR	4.11	0.30	-0.28
TSR	4.56	0.29	-0.25
TNR+TSR	**1.77**	**0.08**	**-0.07**
TNR+TSR−EX	3.05	0.14	-0.10
Miao *et al.* [13]	9.89	0.68	-0.61
SVR	14.78	0.80	N/A
Bayesian regression	13.77	0.84	N/A
WebSphere			
Models	Step Loss (%)	MAE	LL($\times 10^4$)
TNR	4.77	0.40	-0.88
TSR	4.56	0.37	-0.80
TNR+TSR	**1.44**	**0.07**	**-0.19**
TNR+TSR−EX	2.31	0.11	-0.29
Miao *et al.* [13]	7.55	0.60	-0.81
SVR	18.20	0.71	N/A
Bayesian regression	17.02	0.80	N/A
AIX			
Models	Step Loss (%)	MAE	LL($\times 10^4$)
TNR	4.46	0.37	-0.41
TSR	4.15	0.30	-0.35
TNR+TSR	**1.99**	**0.15**	**-0.17**
TNR+TSR−EX	3.86	0.25	-0.25
Miao *et al.* [13]	11.10	0.81	-1.21
SVR	15.08	0.77	N/A
Bayesian regression	12.56	0.85	N/A

Table 3: Effectiveness of Routing Models.

6.3.2 Resolution Efficiency

One natural hypothesis is that under the task-specific routing, a task will be resolved quickly, since TSR directly takes into account the next expert's ability to solve the task. We now verify this hypothesis. Recall that the variational parameter γ_i in our model TNR+TSR reflects the mixture weights used by expert e_i when transferring a task. If in γ_i, the sum of the components corresponding to TSR-related routing patterns is larger than that corresponding to TNR-related patterns, expert e_i is regarded as using TSR more to transfer a task; otherwise the expert is using TNR more. Therefore, we can roughly divide experts into two groups: TNR-kind and TSR-kind. After an expert transfers a task, we count the number of remaining experts needed to resolve the task. We respectively summarize the distribution of the remaining expert number when a TSR-kind of experts transfers a task, and that when a TNR-kind of experts transfers a task. Figure 4 shows the results on the DB2 tickets. It clearly verifies the hypothesis. On DB2 tickets, a ticket will likely get solved with one more step when an expert uses TSR to route it. However, if using TNR, a ticket might still need 2 or 3 more steps to get resolved. We obtain an additional implication from Table 3 and Figure 4, that is, TNR+TSR better captures the expert real routing behaviors in a collaborative network while routing based on TSR can lead to more efficient task resolution. Due to space constraints, we omit the results for AIX and WebSphere, which are very similar to that of DB2.

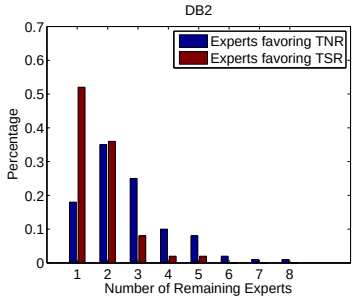

Figure 4: Efficiency of TNR vs. TSR.

6.3.3 Expertise Difference Routing Pattern

In our EX routing pattern, given the expertise of e_i and e_j, we estimate f_{ij}, i.e., the general trend of e_i sending a task to e_j, based on a log-normal distribution of $\Delta(e_i, e_j)$. The selection of log-normal is due to the non-negative nature of $\Delta(e_i, e_j)$ and the asymmetric shape of the distribution shown in Figure 2. However, one might consider estimating f_{ij} based on a normal distribution, since a normal distribution also seems to be quite similar to Figure 2:

$$f_{ij} \propto e^{-\frac{[\Delta(e_i,e_j)-\mu]^2}{2\sigma^2}} \tag{25}$$

Table 4 empirically justifies our selection of the log-normal distribution. (TNR+TSR)$^{\#}$ is the result corresponding to using the normal distribution to estimate f_{ij} based on Eqn. 25, which is much worse compared with TNR+TSR.

Models	Step Loss (%)	MAE	LL($\times 10^4$)
DB2			
TNR+TSR	**1.77**	**0.08**	**-0.07**
(TNR+TSR)$^{\#}$	3.54	0.22	-0.18
(TNR+TSR)*	1.90	0.10	-0.08
WebSphere			
TNR+TSR	**1.44**	**0.07**	**-0.19**
(TNR+TSR)$^{\#}$	3.67	0.24	-0.55
(TNR+TSR)*	1.59	0.08	0.20
AIX			
TNR+TSR	**1.99**	**0.15**	**-0.17**
(TNR+TSR)$^{\#}$	4.01	0.34	-0.38
(TNR+TSR)*	2.12	0.17	-0.19

Table 4: Variants of EX Routing Pattern.

Instead of optimizing μ and σ^2 during model solution, we could pre-estimate μ and σ^2 based on the distribution of the relative expertise difference in the training dataset, as shown in Figure 2, and keep them fixed during model training. We denote this setting as (TNR+TSR)*. As discussed in Section 3.1, the histogram in Figure 2 is due to the integrated effects of all the routing patterns, whereas TNR+TSR optimizes μ and σ^2 with more emphasises on tasks transferred following the EX pattern, and can further improve the accuracy. Nevertheless, given that the performance does not differ too much between TNR+TSR and (TNR+TSR)*, in practice, one might consider saving the trouble of deriving

complicated derivatives over μ and σ^2 during model solution.

6.3.4 Optimizing Collaborations

In the management of real collaborative networks, system administrators need to optimize the current network, e.g., in terms of expert training, to improve the efficiency of task execution. However, it is very expensive, if not impossible, to alter the real collaborative network just for hypothesis testing. Currently such decisions are manually made by experienced managers or consultants, without much quantitative analysis on how the resulting network will perform. Since our model accurately captures the routing behaviors of experts, it can naturally serve as a trustable simulation means for real task routing in the collaborative network. Hypothesises on whether a certain change to the network can improve the efficiency or not, can be much more easily examined with the help of our model.

Here we study optimization of the collaborative network, in terms of training experts to have more efficient routing patterns. Particularly, we examine two questions: *What kind of routing patterns might bring better resolution efficiency? Which expert(s) should be selected for more training, given a limited budget?* Section 6.3.2 implies if an expert is more likely to route a task based on TSR, the task will be resolved more quickly. We now formally verify this hypothesis. For each expert, we treat TSR and TNR as two groups of routing patterns, and set the group mixture weights respectively as $(x, 1-x)$. When an expert transfers a task, we first randomly select TSR or TNR based on the group mixture weights, and then select a routing pattern inside each group according to their mixture weights previously learnt in our model TNR+TSR. According to Section 5, we estimate the completion time of a task, and evaluate the task resolution efficiency by the average CT of all the tasks. We vary x to test the change of the task resolution efficiency. Only the results in the DB2 tickets are shown in Figure 5 since similar results on WebSphere and AIX are also observed. We can see that as the mixture weight for TSR gets closer to 1, the average completion time tends to become smaller, indicating task resolution becomes more efficient.

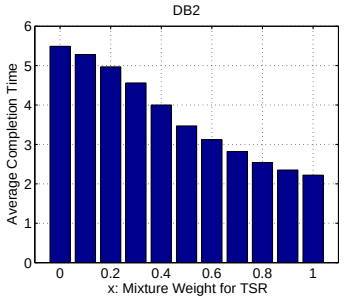

Figure 5: Effect of TSR Weights.

When the training budget is limited, which experts should be trained first to maximize the performance of the entire network is an interesting problem. For simplicity, we consider the problem of selecting the best candidate. One can extend to top-k candidate selection by adopting a greedy method. Possible methods to recommend an expert include (1) randomly select one expert from the network, denoted as *Random*; (2) select the expert that transfers the most tasks, denoted as *Frequent Transferrer*; (3) select the expert that is the least efficient: after the expert transfers a task, the average number of remaining steps to solve a task is the highest, denoted as *Least Efficient*; (4) use our model to conduct task routing after an expert's routing pattern is changed and select the expert that can lead to the most improvement of efficiency. Efficiency improvement is evaluated by the decrease of the average CT of tasks. Methods (2)-(4) are executed on a training task set and evaluated by calculating the efficiency improvement in a testing task set. Table 5 clearly demonstrates that compared with other methods, training the expert recommended with the help of our model, can result in a much more efficiency improvement. This result is expected because through routing a training set of tasks with our model, we are able to know which expert's routing pattern plays a critical role in decreasing the average CT of tasks. This study demonstrates that our model could help conduct hypothesis testing easily and can provide valuable recommendations to decision makers during the optimization of a collaborative network.

Methods	Efficiency Improvement (%)
Random	0.27
Frequent Transferrer	0.91
Least Efficient	1.21
Recommendation with Our Model	**2.75**

Table 5: Training Recommendation

7. RELATED WORK

Our work is related to previous studies in three categories: (1)Collaborative networks; (2)User behavior modeling; and (3)Multi-class classification and Markov processes.

Collaborative Networks. As mentioned in Section 1, Shao *et al.*[19], Miao *et al.*[13], and Zhang *et al.* [23] propose automated routing algorithms to resolve a certain task as fast as possible. The task resolution problem is also related to the expert finding problem[3, 9, 18]: Given a keyword query, find the most knowledgeable persons regarding that query. All of them aim at proposing algorithms that can speed up the resolution of a task or a query. Our work differs from these studies: We are not aimed at building another recommendation algorithm for finding a right resolver. Instead, we try to uncover the patterns underlying human real routing decisions. Miao *et al.* [14] study a network model and a routing model to jointly simulate the structure and the ticket routing procedure in a collaborative network, while we directly infer the routing models in real collaborative networks. A salient feature of our model is its capability of estimating the completion time of a real ticket.

User Behavior Modeling. Information propagation, as one type of user behaviors, has been widely studied, such as [11] on influence maximization, [20] on propagation through e-mail forwarding , [21] on information spreading patterns in Twitter, and so on. Unlike information propagation from person to person, the purpose of task routing in a collaborative network is to find the resolver for the task instead of influencing others. Another type of research studies, *e.g.*,

[5, 10], focus on the measurement of user behaviors. Benevenuto *et al.* [5] study the clickstream data to reveal key features of user behaviors, such as how often users connect to a social network and the sequence of activities users conduct on a social network site. [22] investigates the retweeting behaviors of users in Twitter. Zhong *et al.* [24] leverage the knowledge of user rating behaviors in multiple social networks to enhance the predictive performance of user modeling. Different from these previous studies, in this work, we analyze experts' routing behaviors in collaborative networks: Given a task, how an expert decides where to transfer it and what kind of routing patterns an expert possesses.

Multi-class Classification and Markov Processes. In terms of methodology, our model is related to multi-class classification [17] and Markov processes [4]. When an expert considers where to route a task, the decision process can be regarded as a multi-class classification problem. One can potentially build a classifier for each expert, to decide where to route a task (*i.e.*, the label). It requires effective features and training algorithms to achieve good performance. Proposing such features for classification is not an easy problem. Actually, the routing patterns in our model can be regarded as relevant features for classification. The formalization of a mixture model gives an intuitive explanation of the decision making process of an expert. The task routing problem is also related to Markov process, particularly, Markov chain. States in a Markov chain correspond to experts processing a task at each step in our model. Given the current expert e_i, the next expert to access a task is independent from the experts previous to e_i, which can be regarded as the Markov Property. Different from a classic Markov chain, in our case, the state (*i.e.*, expert) transition probability is with respect to a specific task and is obtained through a mixture of multiple routing patterns.

8. CONCLUSION

In this paper, we model the decision making and cognitive process of an expert during task routing in collaborative networks. A routing decision of an expert is formulated as a result of a generative process based on multiple routing strategies. We formalize each routing strategy in a probabilistic framework, and modeled experts' routing decision making through a generative model. Our analytical model has been verified that it only explains the real routing sequence of a task very well, but also accurately predicts a task's completion time in the current collaborative network. In comparison with all the alternatives, our method improves the performance by more than 75% under three different quality measures. We also demonstrated that our model can provide guidance on optimizing the performance of collaborative networks.

9. REFERENCES

[1] Bugzilla: http://www.bugzilla.org/.

[2] http://en.wikipedia.org/wiki/ibmwebsphere.

[3] K. Balog, L. Azzopardi, and M. De Rijke. Formal models for expert finding in enterprise corpora. In *SIGIR*, pages 43–50. ACM, 2006.

[4] R. Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[5] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida. Characterizing user behavior in online social networks. In *SIGCOMM on IMC*, pages 49–62. ACM, 2009.

[6] C. M. Bishop and N. M. Nasrabadi. *PRML*, volume 1. springer New York, 2006.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[8] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[9] H. Fang and C. Zhai. Probabilistic models for expert finding. In *AIR*, pages 418–430. Springer, 2007.

[10] L. Gyarmati and T. A. Trinh. Measuring user behavior in online social networks. *Network, IEEE*, 24(5):26–31, 2010.

[11] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, pages 137–146. ACM, 2003.

[12] D. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[13] G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, and N. Anerousis. Generative models for ticket resolution in expert networks. In *SIGKDD*, pages 733–742. ACM, 2010.

[14] G. Miao, S. Tao, W. Cheng, R. Moulic, L. E. Moser, D. Lo, and X. Yan. Understanding task-driven information flow in collaborative networks. In *WWW*, pages 849–858. ACM, 2012.

[15] T. M. Mitchell. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45, 1997.

[16] H. Moskowitz and K. Tang. Bayesian variables acceptance-sampling plans: quadratic loss function and step-loss function. *Technometrics*, 34(3):340–347, 1992.

[17] J. C. Platt, N. Cristianini, and J. Shawe-taylor. Large margin dags for multiclass classification. In *NIPS*, pages 547–553, 2000.

[18] P. Serdyukov, H. Rode, and D. Hiemstra. Modeling multi-step relevance propagation for expert finding. In *CIKM*, pages 1133–1142. ACM, 2008.

[19] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis. Efficient ticket routing by resolution sequence mining. In *SIGKDD*, pages 605–613. ACM, 2008.

[20] D. Wang, Z. Wen, H. Tong, C.-Y. Lin, C. Song, and A.-L. Barabási. Information spreading in context. In *WWW*, pages 735–744. ACM, 2011.

[21] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts. Who says what to whom on twitter. In *WWW*, pages 705–714. ACM, 2011.

[22] Z. Yang, J. Guo, K. Cai, J. Tang, J. Li, L. Zhang, and Z. Su. Understanding retweeting behaviors in social networks. In *CIKM*, pages 1633–1636. ACM, 2010.

[23] H. Zhang, E. Horvitz, Y. Chen, and D. C. Parkes. Task routing for prediction tasks. In *AAMS-Volume 2*, pages 889–896. IFAAMS, 2012.

[24] E. Zhong, W. Fan, J. Wang, L. Xiao, and Y. Li. Comsoc: adaptive transfer of user behaviors over composite social network. In *SIGKDD*, pages 696–704. ACM, 2012.