Overlapping community detection and temporal analysis on Q&A sites

Zide Meng, Fabien Gandon, Catherine Faron Zucker

Web Intelligence 15 (2017) 115–142
DOI 10.3233/WEB-170356

Overlapping community detection and temporal analysis on Q&A sites

Zide Meng [a], Fabien Gandon [a,*] and Catherine Faron Zucker [b]
[a] *INRIA Sophia Antipolis Méditerranée, Sophia Antipolis, France*
E-mails: zide.meng@inria.fr, fabien.gandon@inria.fr
[b] *University Nice Sophia Antipolis, CNRS, I3S, France*
E-mail: faron@unice.fr

Abstract. In many social networks, people interact based on their relationship network. Community detection algorithms are then useful to reveal the sub-structures of a network. Identifying these users' communities can help us assist their life-cycle. However, in certain kinds of online communities such as question-and-answer (Q&A) sites or forums, people interact based on common topics of interest, rather than an explicit relationship network. Therefore, many traditional community detection techniques do not apply directly. Discovering those topics of interest is critical to identify users' communities. Besides, users' activities on certain topics of interest are evolving with time and it is therefore very important to extract their temporal dynamics.

In this paper, we first propose Topic Trees Distributions (TTD), an efficient approach for extracting topics from Q&A sites in order to detect overlapping communities. We then extend TTD to propose Temporal Topic Expertise Activity (TTEA), a graphical probabilistic model to extract both topics-based expertise and temporal information. We evaluated and compared our models with state-of-the-art approaches on a dataset extracted from the popular Q&A site StackOverflow.

Keywords: Overlapping community detection, topic model, temporal analysis, probabilistic graphical model, community question answering

1. Introduction

Question-and-answer sites (Q&A sites) initially aimed at enabling users to ask questions to a community of experts. Since these user-generated contents can be later viewed and searched again, people with the same or similar questions can find answers by browsing or searching the questions that were already answered. On one hand, Q&A sites have become huge repositories of question-answer content which support highly valuable and highly reusable knowledge [3]. On the other hand, Q&A sites also contain a large number of users who keep contributing questions and answers. And most of them are more likely to ask questions on topics they are interested in and answer questions in topics they are experts of.

Therefore, we believe that there are two main resources in Q&A sites: the users' network and the Q&A content. From a user's perspective, detecting communities of interests is useful to reveal the sub-structures of the user network and identify relevant peers. From the perspective of content, extracting topics is required to uncover the key subjects from massive content.

So we are interested in the following three research questions: (1) *How can we identify the common topics binding users together?* (2) *How can we detect topics-based overlapping communities?* (3) *How can we extract topics-based expertise and temporal dynamics?*

Detecting this information can contribute to the question routing problem [17,36], which is very important in Q&A sites optimization problems, for example, to recommend a question to a user who is active in the corresponding topic and has the expertise needed to answer it. It can also contribute to the community

*Corresponding author. E-mail: fabien.gandon@inria.fr.

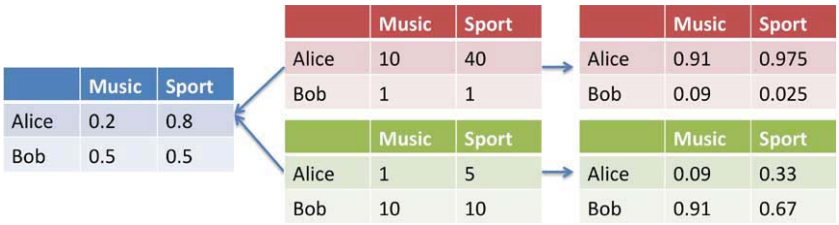

Fig. 1. Different ways to estimate probabilities with results of Gibbs sampling.

management, for instance by allowing to track the interest evolution or community evolution in Q&A sites.

Q&A sites support social networking, however, unlike networks such as Facebook, there are no explicit relationship-based links between their users. In fact, Q&A sites capture the connection of users by question-answer links or co-answer links. The users are neither mainly concerned with nor aware of the links existing between them. The social network is said to be implicit. As a result, compared with other classical social networks, Q&A networks contain more *star-shape* structures (many users linked to a central user) than *triangle-shape* structures (users linked to each other). As a result, many community detection algorithms developed to discover sub-structures in social networks do not apply to Q&A implicit networks. To detect communities of interest in Q&A sites, we consider topic based rather than graph structure based algorithms. Moreover, people may have multiple interests, i.e., they may belong to several communities of interests. It is therefore important to be able to detect overlapping communities.

We first tried to adapt a document clustering algorithm to the user clustering problem, similarly to [18]: we applied the classic LDA (Latent Dirichlet Allocation) [4] algorithm to assign each user into several topic clusters, by replacing in this algorithm the documents by the users, and the document words by the tags acquired by users. The results were encouraging. However, we found three limitations. The first one is that the complexity of this probabilistic model was prohibitive. When analyzing the LDA model, we found that it largely exploits tags' co-occurrence. This inspired us to design a much simpler and faster algorithm to detect topics. Then, based on the detected topics, we were able to identify the users' interesting topics.

The second limitation of original LDA based models is that they are not enough to extract temporal and expertise information. In previous works, [32] only considered modeling the expertise, and [12] only considered modeling the topic trends at a group level. Compared with them, we jointly model both expertise and dynamics and we model topic trends also at the level of the users. This provides more insight on the changes of interest of each user.

The third limitation of LDA based models is as follows. If we use a three-layer LDA model (user-topic-word), this generates two kinds of distributions (user-topic distribution and topic-word distribution) which describe to what extent a user is interested in different topics and to what extent a keyword or a tag is related to a topic. However, as shown in Fig. 1 the same user-topic distribution could be generated by different training data (assume that the hidden variable topic is already generated by Gibbs sampling [9]), which means that the user-topic distribution is incomparable among users. For the upper one, *Alice* is more active in topic *music*, but for the lower one, *Bob* is more active. In order to avoid such problem, we proposed a post processing step for the original LDA model to extract such difference. Experiments show that this post-processing can improve both our model and other LDA variants models' performances when it comes to the task of routing questions in a Q&A sites.

So the main contributions of this paper are: (1) A simple and fast topic detection method to extract topics based on question tags. (2) A simple and fast overlapping community detection method based on a topic model. (3) A joint model to capture topics, expertises, activities and trends. Traditionally, these information have been modeled separately. (4) A post processing step to compare the user-topic distributions generated by LDA based models.

The rest of the paper is organized as follows. Section 2 is a synthesis of state-of-the-art approaches of community detection. In Section 3 we present how we apply the original LDA model to detect overlapping communities. We describe our proposed TTD model in Section 4 and TTD experiments and evaluation in Section 5. We describe our proposed TTEA model in Section 4, and TTEA experiments and evaluation in

Section 7. Finally, we conclude and point out potential future works in Section 8.

2. Related work

2.1. Related work on community detection

We distinguish between three kinds of approaches for community detection, depending on their characteristics: Graph-based methods are based on network structure; Clustering methods are based on the similarity of user profiles; LDA-based methods use probabilistic graphical model.

2.1.1. Graph-based methods

A first and direct solution is to extract an implicit network structure (such as a question-answer network, a co-answer network, etc.) from interaction traces to come down to a traditional community detection problem on social networks. Since intuitively, users are grouped by interests, and most of their interactions are based on shared interests, it is reasonable to induce a network structure from these interactions and then run community detection algorithms on the network. Many classical algorithms have been developed such as [2,29]. There are many constraints when adopting these methods. First, they do not take into account node attributes nor link attributes. Take co-answer network as an example, where nodes represent users and links represent users answering the same questions. In case two users are connected, these methods can only detect that they have answered the same questions many times. They cannot indicate whether they have answered questions on the same topic or on different topics. Second, some of the works adopting this approach cannot detect overlapping communities, while other works such as [29] address this problem.

2.1.2. Clustering methods

Community detection can also be envisioned as a clustering problem. By computing similarities between user profiles, one can detect groups according to clustering results. The choice of the similarity metrics is quite important and largely influences clustering results. To find similar interests, the distance between user's interests must be defined and this definition has a strong influence on the clustering results. For instance, we can consider a bag of tags with their weights to represent an interest, then compute the weighted tag distance to define the interest distance between two users. Clustering methods, such as [8,30], group users according to their features. They do not take the network structure into consideration. Moreover, some clustering algorithms normally output hard-partition communities, where a user is assigned to a single interest group. However, in the scenario we are interested in, a user often has more than one interest and should be assigned to more than one group simultaneously. This is a constraint for those hard-partition algorithms. [7] use spectral clustering to detect topics from the graph of tag co-occurrence. Compared to it, our approach is more efficient since we only run spectral clustering on a co-occurrence graph of selected tags (only 10% of all the tags). Besides, [7] does not give any details on how to compute the topic tag distribution and user topic distribution, while we do.

2.1.3. LDA-based models

A third approach consists in using a probabilistic graphical model for both the user profiles and the network structure to solve community detection problem. For example, [34] transform links to binary node attributes, then use a LDA-based model to detect communities. [25] use a LDA-based method on social tagging systems where users label resources with tags, but they do not consider the problem of overlapping community detection. [26] use an extended LDA-based model to analyze academic social networks in order to find expert authors, papers and conferences. A problem of these LDA-based models is that they normally assume soft-membership [31] which means that a user cannot have high probabilities to belong to several communities simultaneously. That is to say that the more communities a user belongs to, the less it belongs to each community (simply because probabilities have to sum to one). Moreover, [20] and [15] also use statistical model to detect overlapping communities. The difference is that LDA-based models normally integrate topic detection which can be used to interpret detected communities while the two above cited methods only detect overlapping communities without any topic information on each detected communities.

2.1.4. Short summary

Table 1 summarizes the main features of the three approaches. Graph-based approaches normally use link information while ignoring node attributes. Some of them cannot detect overlapping communities or provide membership ratios which are weights denoting to what extent a user belongs to a community. Most of these methods cannot identify the topic in each detected community. Clustering approaches use node attributes to group similar users. Some of their results are hard-partition communities, with no overlapping and

Table 1
Comparison of our method to state-of-the-art approaches

	Uses nodes	Uses links	Overlap	Membership	Topic
Graph-based	no	yes	few	few	no
Clustering methods	yes	no	few	few	no
LDA-based	yes	yes	yes	yes	yes
Our-method	yes	yes	yes	yes	yes

no membership information. LDA-based models overcome the shortcomings of graph-based and clustering approaches, using both node attributes and link information. Besides, LDA-based models normally combine community detection with topic detection, which could be used to interpret detected communities. Our proposed method is similar to LDA-based methods, in that it also enables to detect overlapping communities and identify the topics at the same time. It differs from LDA-based methods in that it enables to consider a user having high probabilities to belong to several communities simultaneously while these methods normally assume soft-membership [31]. In addition, our proposed method is much simpler and faster than LDA-based methods while preserving the quality of the detection.

2.2. *Related work on temporal modeling and expert detection*

2.2.1. *Topic based temporal modeling*

There is an increasing research interest for the temporal modeling of topics and several methods have been proposed. [27] jointly model topics and time label by assuming that words and time stamps are both generated by latent topics. [33] proposed a PLSA-based [11] model to separate temporary topics from stable topics. [12] jointly model latent user groups and temporal topics to detect group-level temporal topics.

2.2.2. *Related work on expert detection*

Research works related on expert identification in Q&A sites are mainly based on link analysis and topic modeling techniques. The general purpose of expert detection is normally to support the question routing task which essentially consists in finding the most relevant experts to answer a newly submitted question. [35] propose a modified PageRank algorithm to rank users in a specific domain. Besides, they propose the Z-score measure to evaluate expertise. Compared with simple statistical measures, for instance the number of best answers provided by a user, the Z-score measure uses both the number of questions and the number of answers posted by a user. Similarly, [14] use the HITS algorithm to discover authorities users. [5] propose a model based on Indegree, which is the number of best answers provided by users, to discovery experts. [17] propose a probability model to estimate users' expertise for the question routing task.

Rather than detecting global experts, another kind of works uses topic models to detect experts at the topic level. [10] proposed a generative model by leveraging the category information of questions on certain Q&A sites. [32] jointly model topics and expertise by integrating a Gaussian Mixture Model to capture vote information. [7] propose a spectral clustering based topic model. [19] propose a generative model to model the triple role of users (as askers, answerers, and voters). Our contribution extends this kind of works.

There are also works applying machine learning techniques to perform expert detection. [13] combine topic models outputs and statistical features and apply a pair-wised learning to obtain a ranked model and recommend expert users for a question. [23] apply machine learning algorithms to identify experts from their early behavior. [3] perform an in-depth study of StackOverflow[1] and show that expert users tend to answer questions more quickly and gain high reputation by higher activity. Their approach is based on features extraction and machine learning algorithms to predict whether a question has a long-term value and whether a question has been sufficiently answered. Their results show that votes information can indicate a user's expertise level while currently, these kind of work normally rely on the outputs of topic models.

Compared with these latest works, our model captures both topics and expertise, and temporal modeling. It can detect topic dynamics both at the global level and at the individual user level. Besides, we propose a post-process method to solve a common problem in LDA based models.

3. Overlapping community detection based on the LDA model

3.1. *Principle of our approach*

In Natural Language Processing (NLP), Latent Dirichlet Allocation (LDA) [4] is a classical docu-

[1] http://www.stackoverflow.com/

ment clustering method. It is used to detect latent topics from documents by constructing a document-topic-word three layer probabilistic graphical model. In this three layer model, document and word can be observed from dataset, while topic is a hidden layer, which could be estimated by the observed data.

In StackOverflow, a user submits a question, then assigns 1∼5 tags to indicate the key points of this question. Other users who are interested in the question may provide answers to the question or comments to other answers. The basic social graph in StackOverflow is a question-answer graph. As tags attached to a question can reflect the boundary or domain of it, users answering the question can be considered as interested by this domain. As a result, a basic approach is as follows. A user answering a question acquires tags attached to this question. Gradually, each user acquires a list of tags associated with their frequencies. If we treat a user as a document, tags acquired by the user as words in a document, then community detection could also be considered as a clustering problem. User with similar topics of interest are partitioned into the same cluster as a community of interest.

Similarly to [18], we applied the classic LDA method to construct a users-topics-tags model to detect latent topics of interest from the tags acquired by users and then cluster users into different topics. The output of the model consists in two probability distributions: (1) A User-Topic distribution to describe to what extent a user is interested in different topics. (2) A Topic-Tag distribution to describe to what extent a topic is related to different tags.

The formalization of this model is given by equation (1):

$$P(t|u) = P(t|z) * P(z|u) \tag{1}$$

where t denotes a tag, z denotes a latent topic, u denotes a user. The probability of a tag for a user is the result of multiplying the probability of this tag for a topic and the probability of this topic for the user.

The plate notation of this model is presented in Fig. 2.

The dependencies among the many variables can be captured by the direction of line. The boxes represent replicated variables, which are users, interests and tags. The outer boxes represent users, while the inner boxes represent the repeated choice of topics and tags for a user. The parameters of this model are explained in Table 2. M and V are given while K, α and β can be chosen. T is observed in users' tag lists. Other variables are latent variables which have to be estimated.

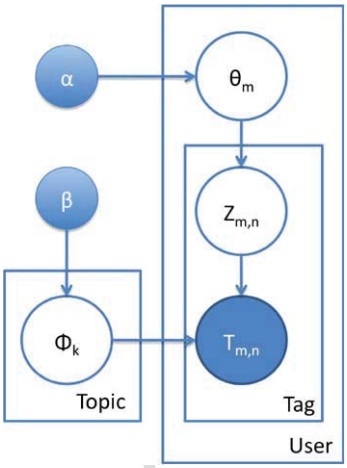

Fig. 2. User–Topic–Tag (LDA) model.

Table 2
Model parameters

Parameter	Meaning
M	the total number of users
K	the total number of topics
V	the total number of tags
N_m	the total number of tags for user m
α	the parameter of the Dirichlet prior on the per-user topic distributions
β	the parameter of the Dirichlet prior on the per-topic tag distributions
θ_m	the topic distribution for user m
ϕ_k	the tag distribution for topic k
$z_{m,n}$	the topic for nth tag in m's tag list
$t_{m,n}$	the specified tag in nth position of m's tag list

The intuition behind this model is that users choose their topics and topics generate tags. The generative process can be summarized as follows:

Process of generating a user's tag list
for interest k **in** $[1..K]$:
  draw topic tag distribution $\phi(k) \sim \text{Dir}(\beta)$
for user $m \in [1, M]$:
  draw a user-topic distribution $\theta(m) \sim \text{Dir}(\alpha)$
  for each tag $n \in$ user m's tag list, where $n \in [1, N_m]$, $m \in [1..M]$
    draw topic $z_{m,n} \sim \text{Multi}(\theta(m))$
    draw tag $t_{m,n} \sim \text{Multi}(\phi(z_{m,n}))$

We use the collapsed Gibbs sampling method [9] to sample the hidden variable z, then θ and ϕ can both be

estimated. The inference process is as follows. We iteratively sample the topic indicator $z_{m,n}$ for each answer tag $t_{m,n}$ according to equation (2).

$$p(z_i = z_{m,n} | u = u_m, t = t_{m,n}, Z, U, T_{\neg i}) \propto \frac{C^{z_{m,n}}_{u_m,\neg i} + \alpha}{\sum_{k=1}^{K} C^{k}_{u_m,\neg i} + K * \alpha} \cdot \frac{C^{t_{m,n}}_{z_{m,n},\neg i} + \beta}{\sum_{t=1}^{V} C^{t}_{z_{m,n},\neg i} + V * \beta} \quad (2)$$

where $\neg i$ enforces that all the counters used are calculated with tag t_i excluded. $C^{k}_{u,\neg i}$ is the number of tags acquired by user u assigned to topic k, $C^{t}_{k,\neg i}$ is the number of tags t assigned to topic k.

Then with a Gibbs sampling, we can estimate θ and ϕ by equations (3) and (4):

$$\theta = \frac{C^{k}_{u} + \alpha}{\sum_{k=1}^{K} C^{k}_{u} + K * \alpha} \quad (3)$$

$$\phi = \frac{C^{t}_{k} + \beta}{\sum_{t=1}^{V} C^{t}_{k} + V * \beta} \quad (4)$$

where C^{k}_{u} is the number of tags assigned to topic k of user u, C^{t}_{k} is the number of tags t assigned to topic k.

3.2. Experiments

We run the above described model on a dataset from the popular Q&A site StackOverflow, each user being represented by her tag list as explained before. We just illustrate some of the results to show the effectiveness of this model.

The first result is the probability for each tag to belong to each topic. This is shown in Table 3. The second result is the probability for a user to belong to different topics of interest. This is shown in Table 4.

Table 3 shows eight detected topics of interest, one column for one topic, and ten rows for the top 10 tags for each topic, sorted by descending weights (a tag's weight is the probability of the tag to belong to the topic). This table shows that each topic has a clear and focused interest. For example, topic 1 has c-development related tags, topic 2 has java-development related tags, topic 3 has c#-development related tags, topic 4 has html-development related tags, topic 5 has iphone-development related tags, topic 6 has database related tags, topic 7 has linux-development related tags, topic 8 has non-programming related tags. Moreover, weights reflect the relevance of tags to each topic. For example, topic 5 is

Table 3
Top 10 related tags for detected topics of interest

Topic 1	Topic 2	Topic 3	Topic 4
c++ (0.225)	**java** (0.345)	**c#** (0.225)	**php** (0.117)
c (0.084)	**eclipse** (0.023)	**.net** (0.128)	**javascript** (0.115)
windows (0.020)	**swing** (0.015)	**asp.net** (0.059)	**html** (0.059)
stl (0.014)	best-practices (0.014)	**vb.net** (0.019)	**jquery** (0.056)
algorithm (0.014)	multithreading (0.011)	**linq** (0.018)	**css** (0.042)
c# (0.013)	xml (0.010)	windows-forms (0.016)	mysql (0.029)
win32 (0.013)	**spring** (0.010)	**visual-studio** (0.015)	**ajax** (0.021)
linux (0.011)	performance (0.009)	**asp.net-mvc** (0.015)	**web-development** (0.019)
best-practices (0.011)	jsp(0.008)	wpf (0.012)	regex (0.018)
multithreading (0.011)	generics (0.008)	best-practices (0.011)	**asp.net** (0.015)
Topic 5	**Topic 6**	**Topic 7**	**Topic 8**
iphone (0.137)	**sql** (0.181)	**python** (0.181)	subjective (0.143)
objective-c (0.123)	**sql-server** (0.150)	**perl** (0.056)	**best-practices** (0.038)
cocoa (0.080)	**database** (0.062)	**regex** (0.031)	**language-agnostic** (0.035)
ms-access (0.062)	delphi (0.042)	**linux** (0.030)	programming (0.028)
cocoa-touch (0.056)	**sql-server-2005** (0.042)	**ruby** (0.027)	**not-programming-related** (0.019)
iphone-sdk (0.041)	**mysql** (0.039)	django (0.023)	**career-development** (0.018)
vba (0.035)	**tsql** (0.037)	ruby-on-rails (0.021)	**learning** (0.017)
excel (0.023)	**oracle** (0.028)	beginner (0.017)	polls (0.017)
vb6 (0.022)	**database-design** (0.025)	git (0.013)	programming-languages (0.015)
xslt (0.021)	**stored-procedures** (0.017)	**bash** (0.013)	**design** (0.014)

Table 4
Detected topics of interest

user_21886	user_14860	user_15401
html-development (0.284), **c-development** (0.275)	**c-development** (0.333), **linux-development** (0.196)	**database-related** (0.383), **non-programming-related** (0.290)
python (93)	c (152)	sql-server (108)
c++ (64)	c++ (148)	database (64)
javascript (45)	java (89)	sql (63)
html (34)	subjective (89)	subjective (45)
c# (33)	c# (68)	python (43)
css (32)	sql (68)	sql-server-2005 (31)
visual-studio (29)	windows (67)	best-practices (27)
windows (27)	linux (54)	.net (25)
c (27)	bash (48)	c++ (23)
.net (24)	regex (43)	c# (22)
user_78374	**user_53897**	**user_23743**
non-programming-related (0.493), **linux-development** (0.316)	**java-development** (0.835), **non-programming-related** (0.075)	**iphone-development** (0.683), **non-programming-related** (0.155)
subjective (35)	java (366)	objective-c (73)
python (32)	eclipse (24)	cocoa (71)
best-practices (16)	tomcat (20)	iphone (34)
c (13)	subjective (18)	cocoa-touch (21)
programming (13)	performance (18)	mac (19)
c++ (10)	best-practices (16)	osx (17)
beginner (8)	j2ee (14)	iphone-sdk (13)
not-programming-related (8)	jar (13)	xcode (10)
language-agnostic (6)	logging (10)	subjective (8)
coding-style (5)	c# (9)	c (8)

concerned with iphone-development, its top 3 tags are 'iphone', 'objective-c' and 'cocoa' which are very relevant to it.

Table 4 shows six randomly chosen users and their top 10 tags. The first row contains user ids, the second row contains their detected topics of interest with their probability. The following ten rows show the top 10 tags for each user. We replaced topic ids with topic names which we have assigned to them according to their associated tags.

3.3. Discussion

The above experiments show that, by applying topic models on Q&A website, we are able to detect overlapping communities, and the detected topics are useful to explain each corresponding community. In our work, we directly use each topic to represent a community.

However, we found that there are three limitations when applying LDA models to our task. The first one is a lack of efficiency: the complexity of the probabilistic model was prohibitive. The second limitation is that the original LDA model does not enable to extract temporal and expertise information. The third limitation is that the detected probability distributions cannot be compared with each other. Therefore, we extended our work in two directions. First, we developed a simpler method to detect topics and overlapping communities to solve the first problem: the TTD method is presented in Section 4. Second, we propose a more complex model to extract more information from user generated content to answer the two other limitations: the TTEA method is presented in Section 6.

4. Topic trees distributions (TTD)

4.1. Problem definition

In StackOverflow, a user submits a question, then assigns 1∼5 tags to indicate the topics of the question. Other users who are interested in the question may provide answers to the question. As the tags attached to a question can reflect its topic, users answering a question can be considered as interested by its topic. Let $U = \{u_1, u_2, \ldots, u_n\}$ be the set of users, $Q = \{q_1, q_2, \ldots, q_m\}$ the set of questions and $T = \{t_1, t_2, \ldots, t_v\}$ the set of tags. We aim at (1) extracting topics distribution $Topic = \{topic_1, topic_2, \ldots, topic_k\}$ from T, and for each $topic_k$, defining $topic_k = \{p_{k1}, p_{k2}, \ldots, p_{kv}\}$ where p_{ki} denotes the probability of tag t_i to be related to $topic_k$; and then (2) detecting user's interests. For a user $u_i \in U$, we define $I_i = \{I_{i1}, I_{i2}, \ldots, I_{ik}\}$ where I_{ik} denotes the probability of u_i to be interested in $topic_k$.

4.2. First-tag enrichment

When sorting the tags of a question by their global frequency, we found that normally the first tag of a question is much more general and indicates the domain of the question. For example, a question tagged with {*c#*, *iostream*, *fstream*} is related to *c#*; a question tagged with {*html*, *css*, *height*} is related to *html*. However, there are also some questions which have less tags and, in this case, the tags are less popular,

Fig. 3. Example of computing a first-tag distribution.

like a question tagged with {*ant*} or a question tagged with {*qt*, *boost*}. For these questions, the main domain is implicit. Our experiment dataset shows that nearly 12% of the questions only have one tag, and nearly 25% of the questions only have two tags. Therefore, we propose an approach to enrich a question with a first tag when needed. The first step of our approach consists in computing the first-tag distribution. For example, as shown in Fig. 3, let us consider the three tag lists, {*html*, *css*, *height*}, {*html*, *css*, *layout*}, and {*c#*, *gui*, *layout*}, respectively associated to questions Q1, Q2, Q3. The first-tag frequency map for *html* is {*html*:2}, the first-tag frequency map for *css* is {*html*:2}, and the first-tag frequency map for *layout* is {*html*:1, *c#*:1}. Given a tag, the probability of its first-tag is computed by equation (5), which is the Maximum Likelihood Estimation (MLE) of the probability $p(first\_tag|tag)$, where $I(tag)$ denotes the occurrence of *tag* and $I(first\_tag, tag)$ denotes the co-occurrence of *first_tag* and *tag*.

$$p(first\_tag|tag) = \frac{p(first\_tag, tag)}{p(tag)} = \frac{I(first\_tag, tag)}{I(tag)} \quad (5)$$

We compute the probabilities just by normalizing the first-tag frequency map. In the example, the first-tag frequency map for *css* becomes {*html*:1.0} and the first-tag frequency map for *layout* becomes {*html*:0.5, *c#*:0.5}. In order to lower the probabilities of low frequency tags as first-tag, we use the squashing function (6):

$$p(first\_tag|tag) = \frac{I(first\_tag, tag)}{I(tag)} * \sigma\big(I(first\_tag)\big) = \frac{record\_freq}{\text{sum}(record\_freq)} * \frac{1}{(1 + e^{-k*freq})} \quad (6)$$

where, *record_freq* denotes the co-occurrence of *first-tag* and *tag*, sum(*record_freq*) denotes the sum of these recorded frequencies, *freq* denotes the global occurrence of the first-tag, $\sigma(x)$ is sigmoid function, which is used as a squashing function for numerical stability. The value of sigmoid function is between 0 and 1, however the shape of this function is largely determined by parameter k. Considering the maximum value of tag frequency (tag c#:31,801) in our dataset, we chose k as 0.001 (dotted line), which will lower the probabilities of low frequency tags as first-tag while maintaining the probabilities of high frequency tags as first-tag. Figure 4 recalls the shape of the sigmoid function for different values of k.

For example, if the first-tag frequency map for *css* is {*html*:10, *jquery*:2}, then, when normalizing first-tag *html*, $record\_freq = 10$, $\text{sum}(record\_freq) = 12$, $p(html) = 5{,}552$. As a result, $p(html|css) = 0.8301$. Similarly, for each tag, we provide a list of enriching first-tags with estimated probabilities.

The second step of our approach consists in choosing a first-tag to enrich each question. Given a question's tag list, we fetch the top 5 first-tags (with the highest probabilities). Then we accumulate the corresponding probabilities with a discount taking into account the position of the tag in the tag list associated to the question, as shown in equation (7):

$$p_j = p_{1,j} + p_{2,j} * dis + \cdots + p_{k,j} * dis^{k-1} \quad (7)$$

where p_j denotes the probability of tag j to be the first-tag of a given question, $p_{k,j}$ denotes the probability for tag k to have tag j as its first-tag. The range of v and k are $[1, V]$ and $[1, K]$, where V denotes the number of all the first-tags, K denotes the number of tags in the given question and *dis* denotes the discount due to the position.

Then we consider the first-tag with the highest probability as the enriching first-tag. If this first-tag already

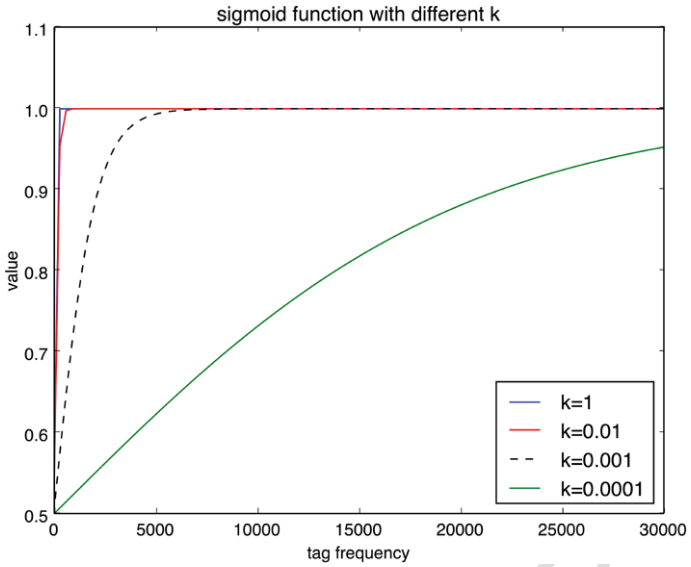

Fig. 4. Shape of function $\frac{1}{(1+e^{-k*z})}$ for different values of k.

Table 5
Original and enriched tag lists

ant	**java**, ant
qt, boost	**c++**, qt, boost
django, hosting	**python**, django, hosting
xslt, dynamic, xsl	**xml**, xslt, dynamic, xsl
sql-server-2005, sorting	**sql**, sql-server-2005, sorting
tomcat, grails, connection	**java**, tomcat, grails, connection
cocoa, osx, mac, plugins	**objective-c**, cocoa, osx, mac, plugins
spring, j2ee, module, count	**java**, spring, j2ee, module, count

exists in the original tag list, we simply skip the insertion, or else we insert it at the first position of the question's tag list. We processed 242,552 tag lists from the StackOverFlow Q&A site, and our method enriched 33,622 of them (13.5%). Table 5 presents the results of the enrichment of 8 tag lists (enriched tags are in bold).

4.3. Topic extraction

From the observation of our dataset, we confirmed the natural intuition that high frequency tags are more generic and low frequency tags are more specific, and most of the low frequency tags are related to a more generic tag. A similar observation was also found in [21]. Besides, [32] shows that tag frequency in Q&A sites also satisfies a power law distribution [1]. For example, for a question tagged with {*c++*, *iostream*, *fstream*} (with tags sorted according to their frequencies), we could find that it was related to *c++* and to the *iostream* topic of *c++*, and more specifically, that it focused on *fstream*. This inspired us to build a tag tree to represent it and compute the probability for a tag to be related to a topic. Figure 5 illustrates the process of building a tag tree. Figure 6 illustrates an example of *html*'s tree. Our topic extraction method is described in Algorithm 1. In the *build trees* process (lines 3–6), we build a tag tree according to the position of tags in a question, and record the occurrence of each node. For example, let us consider again the tag lists of questions Q1, Q2, Q3 in Fig. 3. Based on them, we construct two trees. The root of the first tree is *html*, the occurrence of this node is *2*, it has only one child *css*, which has *2* occurrences, and this node has two children, *layout* and *height*, and each one occurs *1* time. The root of the second tree is *c#* with *1* occurrence. By processing all the tag lists, many trees are generated. We then construct an affinity matrix of the root nodes (lines 7–9). Since we applied our first-tag enrichment method, the number of root tags is not very large. The similarity of two root nodes is computed according to equation (8):

$$\text{Simi}(root\_i, root\_j) = \frac{I(root\_i, root\_j)}{(I(root\_i) + I(root\_j))} \tag{8}$$

where $I(root\_i, root\_j)$ denotes the co-occurrence of tag $root\_i$ and tag $root\_j$, and $I(root\_i)$ and $I(root\_j)$ denote the occurrence of tag $root\_i$ and tag $root\_j$ respectively. Then we perform a spectral clustering [22]

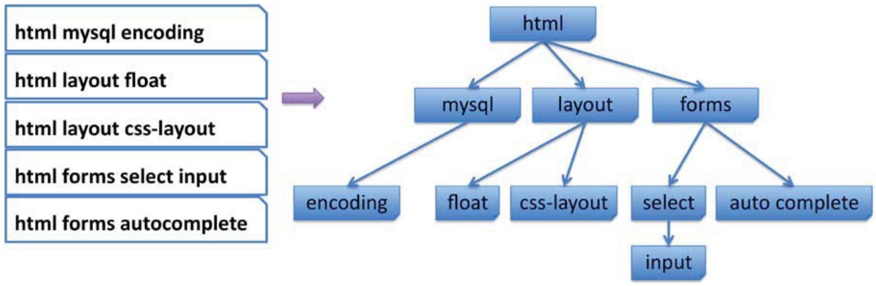

Fig. 5. Example of a tag tree.

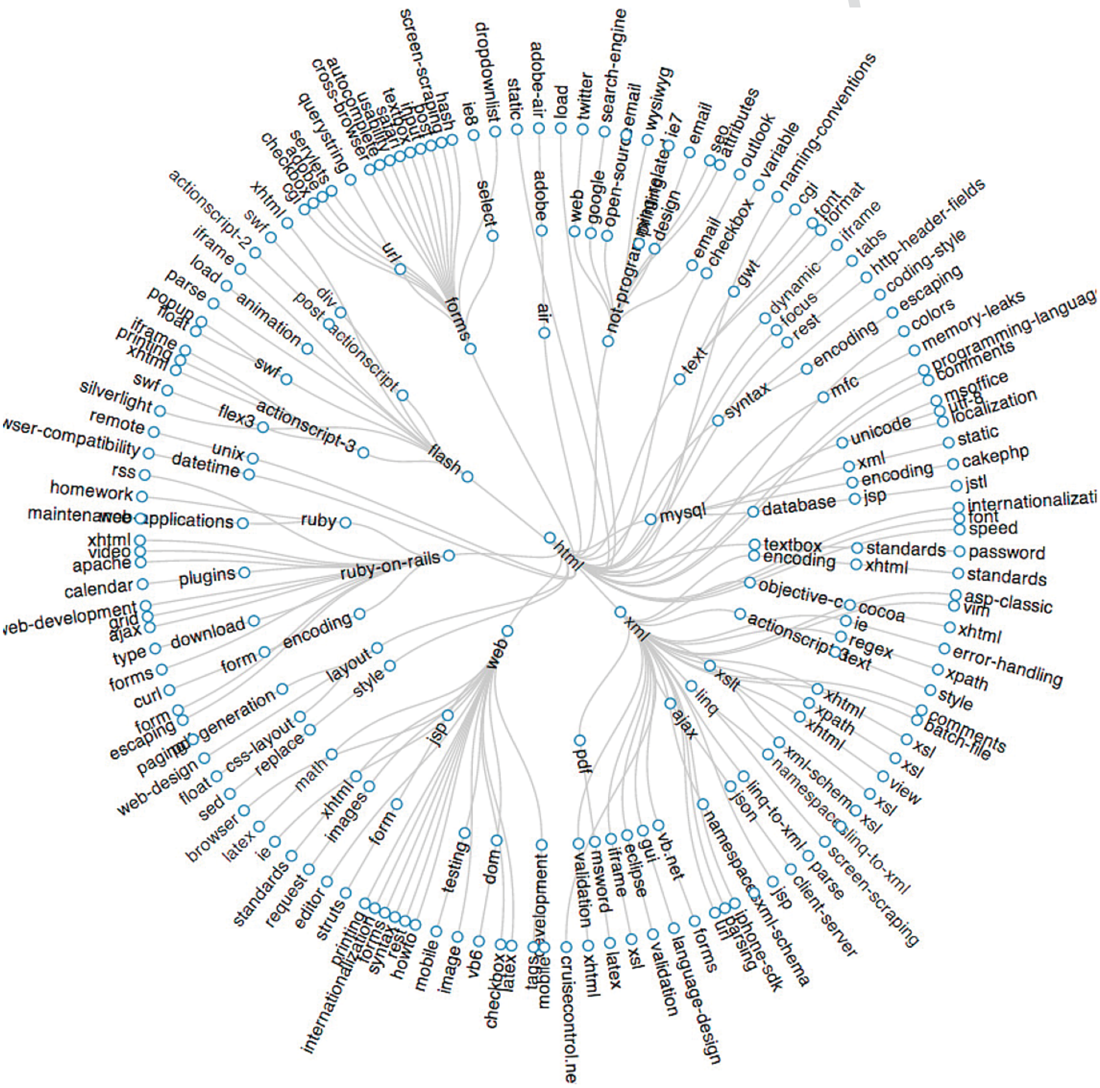

Fig. 6. *html*'s tag tree.

Algorithm 1 Topic Extraction

```
1: Input: enriched tag list of questions, topic number K
2: Output: topic-tag distribution
3: /*build trees process, shown in Fig. 5*/
4: trees = null /* initialize */
5: for tag in taglist do
6:     trees.insert(taglist)
7: /*build affinity matrix for root_tags*/
8: root_tags = trees.get_root_tags()
9: affinities_matrix = build_affinity(root_tags)
10: /*run spectral-clustering on affinity matrix*/
11: groups = spectral(affinities_matrix,K)
12: /*combine tree according to groups*/
13: new_trees = combine_tree (trees,groups)
14: /*compute topic-tag distribution*/
15: topic_distributions = compute (new_trees)
16: ** we perform a spectral clustering to divide these
    root tags into several groups
```

on the affinity matrix to group these root nodes (lines 10–11). Each group forms what we will call a topic. As spectral clustering requires to select the desired number of topics, we choose the same number *30* as [7], which has proved to be a reasonable setting for the Stackoverflow dataset. We then combine trees if their root nodes belong to the same topic (lines 12–13). This process leads to a forest where each tree represents a topic. Then, in the *compute topic-tag distribution* process (lines 14–15),for each topic tree, we compute $p(tag|topic)$ by using the MLE estimation method, according to equation (9):

$$p(tag|topic) = \frac{p(tag, topic)}{p(topic)} = \frac{I(tag) + 1}{I(\text{sum}(tag) + N)} \quad (9)$$

where $I(tag)$ denotes the number of occurrences of *tag* in the topic tree, and $I(\text{sum}(tag))$ denotes the total number of occurrences of all tag occurrences in the topic tree. Compared with LDA-based model, our model could have a zero-probabilities problem, with less popular or new tags related to some topics with a zero probability due to no evidence of co-occurrence. For example, if tag *zombie-process* never occurs in a *html*-related tag tree, then the probability of tag *zombie-process* to be related to *html-related* topics is zero, which could lead to some problems when dealing with young datasets. We avoid it by using the Laplace smoothing method, as shown in equation (9). Table 6 shows the top tags and their probabilities detected by our method.

Table 6
Top tags and their probabilities for some topics computed with TTD

Topic 4		Topic 5		Topic 6	
iphone	0.203	git	0.198	sql	0.177
objective-c	0.112	svn	0.096	mysql	0.122
ios	0.109	version-control	0.045	sql-server	0.074
xcode	0.042	github	0.033	database	0.040
cocoa-touch	0.021	tfs	0.033	oracle	0.030
ipad	0.020	maven	0.029	sql-server-2008	0.029
cocoa	0.018	tortoisesvn	0.018	tsql	0.026
uitableview	0.012	msbuild	0.016	query	0.025
ios5	0.010	jenkins	0.015	sql-server-2005	0.019
core-data	0.009	tfs2010	0.014	database-design	0.011
Topic 12		**Topic 13**		**Topic 14**	
html	0.214	javascript	0.264	machine-learning	0.247
css	0.201	jquery	0.114	artificial-intelligence	0.130
xhtml	0.017	html	0.035	neural-network	0.062
web-development	0.016	ajax	0.031	classification	0.046
ie	0.012	css	0.016	data-mining	0.037
css-layout	0.010	firefox	0.013	svm	0.031
div	0.010	dom	0.011	weka	0.025
layout	0.010	php	0.011	libsvm	0.015
firefox	0.009	ie	0.010	nlp	0.024
ie6	0.009	web-development	0.008	bayesian	0.011

We used the spectral clustering implementation of scikit-learn toolkit.[2] We only run it on the root nodes, which have quite a small size (around 1,175 nodes with the tag enrichment process), which means that we only need to build an affinity matrix on these root nodes and the overall cost is acceptable.

4.4. User interest detection

In StackOverflow, users answering a question can be considered as interested in the topics denoted by the tags of the question. As a result, a starting point for user interest detection is to model the initial situation as follows: a user answering a question ac-

[2] Scikit-learn toolkit: http://scikit-learn.org/stable/modules/clustering.html#spectral-clustering.

quires the tags attached to this question and gradually, each user acquires a list of tags. So we represent a user by a tag list: $U = \{U_i | i = 1, \ldots, n\}$, $U_i = \{tag_i | i = m, n, \ldots, k\}$, and our goal is, for each user U_i, to find $I_i = \{I_{i1}, I_{i2}, \ldots, I_{ik}\}$ where I_{ik} denotes the probability of user U_i to be related to $topic_k$. As we already have a topic-tag distribution we simply compute the user-topic distribution according to equation (10) where $P_{t,k}$ denotes the probability of tag t to be related to topic k. We then normalize the probabilities between 0 and 1 by dividing the global max value. We use the log function for numerical stability. Here we do not apply normalization at the level of the user, because like [31], we believe that each user could have a high interest in two or more topics simultaneously, while most of the probabilistic graphical models including LDA and PLSA require that the sum of all the probabilities is 1, which means that a user cannot have high probabilities to many topics simultaneously. Our method does not have this limitation.

Then we identify users' communities of interests based on the user-topic distribution: a user having a high probability for a topic should be a member of the community represented by this topic.

$$I_{i,k} = \log\left\{\sum_{t=1}^{v} P_{t,k} + 1\right\} \tag{10}$$

5. TTD experiments and evaluation on StackOverflow data

We conducted experiments on the dataset of activities on StackOverflow between 2008 and 2009, which is available online,[3] to evaluate the performance of our TTD approach compared to three other community detection algorithms. The total number of users is 103K. Among them, 47K users submitted at least one question, and 54K users answered at least one question. The total number of tags attached to questions is 24K, and 20% of them are used more than 10 times. The frequency of tags follows a power law distribution. The total number of posts is 1.1M; among them there are 242K questions and 870K answers.

[3] https://archive.org/details/stackexchange

5.1. Performance of topic extraction

We use the Perplexity [4] metric to measure the topic extraction performance. It is a common metric in the topic modeling area, measuring how well the words in test documents are represented by the word distribution of extracted topics. The intuition is that a better model will tend to assign higher probabilities to the test dataset, corresponding to a lower perplexity value. We split the dataset (question tag lists), 80% as training set, 20% as testing set. We run LDA and our method on the training set to get the topic distribution. Then for a test set of M questions' tag lists (N_d denotes the number of tags in the dth question) the Perplexity score is computed as shown in equation (11):

$$\text{Perplexity}(D_{\text{test}}) = \exp\left\{-\frac{\sum_{d=1}^{M} \log p(tag)}{\sum_{d=1}^{M} N_d}\right\} \tag{11}$$

In our model, $p(tag)$ is equal to $p(topic|question) * p(tag|topic)$. We compute the topic-question distribution $p(topic|question)$ similarly to the user-topic distribution (see Section 4.4), by replacing user's tag lists by question's tag lists. The only difference is that we normalize the question-topic distribution to make sure that the sum of a question's topic distribution is 1. We show and compare the average perplexity score in Fig. 7. *TTD* is our method, *TTD_noEnrich* represents our method without first-tag enrichment. We find that TTD could outperform the state-of-the-art LDA method. The reason is that, compared with traditional document topic modeling use cases, question tag lists in Q&A sites are very short, and LDA performs poorly in this situation. Besides, our first-tag enrichment method can improve the performance when the number of topics is not very large. Another point is that, benefiting from a tree structure for topics, we can easily extract sub-topics from a given topic. Besides, TTD is based on a topic model, so extracting these sub-topics can help us find sub-communities within a detected community. Table 7 shows the top tags of *java*'s sub-topic *html* and of topic *html*. We can find that the differences are noticeable for topics: a user who is interested in topic *html* is not necessarily interested in *java*'s sub-topic *html* and vice versa.

5.2. Performance of user interest detection

Traditional community detection algorithms are based on a network structure. As there is no explicit

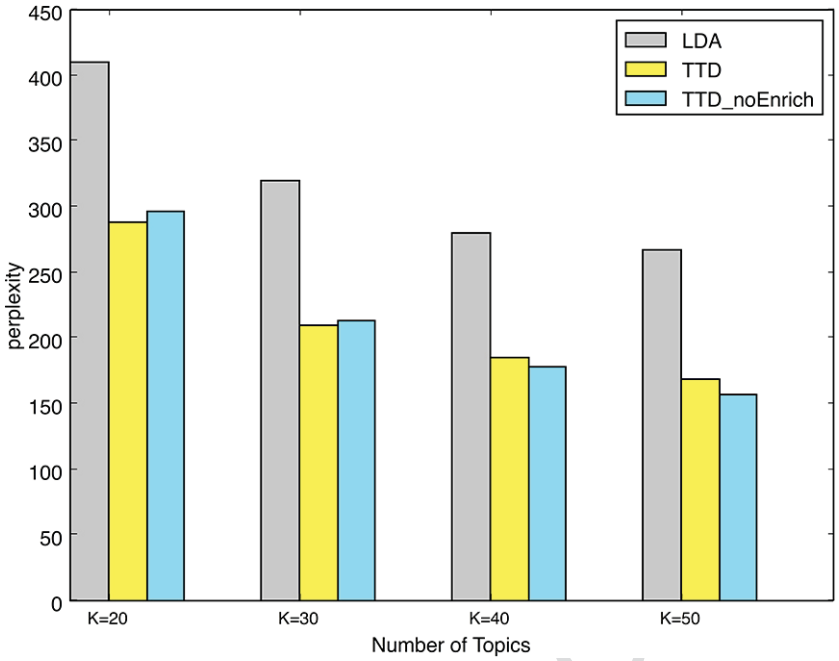

Fig. 7. Comparison of topic extraction performances.

Table 7
Top tags for *java*'s sub-topic *html* and *mysql*, denoted by java_html, and java_mysql respectively, compared with topics *html* and *mysql*

java_html	jsp swing xml parsing jsf jeditorpane pdf applet dom
html	css xhtml web-development table div ie layout css-layout firefox
java_mysql	jdbc hibernate database tomcat prepared-statement spring connection-pooling connection security
mysql	database query mysql-query ruby-on-rails database-design performance stored-procedures innodb optimization

network in our dataset and in order to compare our work with other approaches on the same dataset, we extracted a network of interactions between users: a co-answer network inspired by the notion of co-view network introduced in [8]. The idea behind it is that if two users answer the same question they share some of their interests. So, the co-answer network, to some extent, can reflect the common interests between users. We filtered the co-answer links with a rule stating that a link is kept if two users answer the same questions more than 10 times (we varied this parameter by 15, 20, 25, the results are similar, so here we report results with 10). Based on the noise-less dataset obtained, we implemented three well known community detection methods in order to compare our approach with them. In order to evaluate the results of overlapping community detection, for each user, a method should output 1∼3 community labels with corresponding probabilities to indicate to what extent the user is interested in the community. Then we define three levels of interest in a community: *High*, *Medium*, *Low* according to the probabilities. In addition, we empirically set the number of communities to 30 for all the evaluated methods.

- SLPA [29]: An overlapping community detection method inspired by a classical Label propagation algorithm (LPA). SLPA algorithm can evaluate to which extent a user belongs to a community by the received propagated label (a 'Post-process' in SLPA algorithm). So, it can output more than one community label according to these frequencies.
- LDA: Similar to [32], we run LDA to build a user–topic–tag model on the given dataset, users are represented by their tag list. As the output contains a user–topic distribution, we just sort the distribution for each user and choose the top 3 topic labels as community label together with their probabilities.
- Clustering: We used the implementation of hierarchical clustering from scikit-learn toolkit.[4] As

[4] http://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering

clustering algorithms are hard-partitioned, it can only generate one group label for each user.
– TTD: it is our method. We sort the results of user interest detection (Section 4.4) and choose the top 3 as community label together with their probabilities.

Our aim was to evaluate the similarity between users within a detected community of interest. We mainly used the *Jaccard similarity* and *cosine similarity* of two user's tag lists to evaluate the similarity of two user's interests. We used a modified modularity metric to compute the difference between the average similarity between the users within a community (*avg_inner*) and the average similarity between the users in a community and some user randomly chosen from the whole dataset (*avg_rand*). This is captured in equation (12), where N represents the number of users in a community C, and Simi denotes the similarity function. *Rand_U* represents users that are randomly chosen from the whole data set. A higher value of *avg_inner* denotes that users within a community are very similar. A lower value of *avg_rand* denotes that users of a community are not very similar to random users. So a higher value of *modularity* means a larger difference between *avg_inner* and *avg_rand*, which is considered as a better partition of communities. As the metric has random variables, we run the experiments 10 times and each time we used different random users. Besides, we created a *center* user in each community by averaging all users' tag lists and frequencies, then we computed the average similarity between each user in a community and this *center* user as *avg_center*. As introduced before, each method gives 1∼3 community labels for each user to indicate the level of interest. So we evaluated each level of interest respectively.

$$M(C) = Avg\_inner\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\mathrm{Simi}(U\_i, U\_j)\right) \Big/ Avg\_rand\left(\sum_{i=1}^{N}\sum_{j=1}^{50}\mathrm{Simi}(U\_i, Rand_U\_j)\right) \tag{12}$$

Experiment results are shown in Table 8. We run each method on the co-answer dataset 10 times, and listed the average value. We found that our method is better than the three other methods in detecting users' *High* level of interest with both metrics. The reason why our method is not very efficient to detect users' *Low* level of interest is that our method allows users to belong to more than one community with high probabilities, since our method do not have the sum-to-one constrain. For example, a user could be interested in a topic with a probability of 0.7 (High) and interested in several topics with a probability of 0.3 (Low), then this user will be in many *Low* level of interest communities. This puts some irrelevant users with *Low* level of interest which decreases the similarity between community members. Table 9 shows some users and their interests detected with TTD and their top 10 tags. The first row contains user ids, the second row contains their detected communities of interests with their probabilities. The following ten rows show the top 10 tags for each user. We replaced community labels by names assigned according to the tags associated to each topic of interest.

5.3. Scalability

We also evaluated the scalability of each method. However, as these methods are written in different programming languages, it is not fair to consider this as a precise evaluation; it is just an indication. To increase the stability of the comparison, we run experiments 10 times, and listed the average values. We used a Java implementation of LDA algorithm. All the other methods were implemented in Python. For our method, the time of topic detection was also counted in. For LDA and SLPA, we set the iteration number to 100. We run the experiments on a computer with 3 GHz Intel i7 CPU and 8 GB RAM. From the experiment, we could find that LDA, SLPA and our method are linear in terms of the number of users, see Fig. 8. The LDA algorithm is theoretically $O(nm)$ [28] for each iteration, with n representing the number of users, and m representing the number of tags for each user. However when we test it on large datasets, it clearly appears that only n actually has an impact and m has a very low impact because they are of very different orders. So LDA could be regarded as linear in n. Our method is theoretically $O(nm) + O(r^2)$ without iteration with n representing the number of users, m representing the number of tags for each user, and r representing the number of root tags using our TTD model. Besides, [9] proved that LDA model re-

Table 8
Comparison of the performances of the methods of user interest detection

Similarity	Jaccard similarity											
Level	High interest				Medium interest				Low interest			
Metric	avg_inner	avg_rand	modularity	avg_center	avg_inner	avg_rand	modularity	avg_center	avg_inner	avg_rand	modularity	avg_center
TTD	**0.162**	**0.033**	**4.909**	**0.218**	**0.135**	**0.039**	**3.462**	0.171	0.107	0.042	2.548	0.131
LDA	0.147	0.035	4.200	0.178	0.131	0.039	3.359	**0.177**	**0.144**	0.041	**3.512**	**0.193**
SLPA	0.131	0.040	3.275	0.166	0.129	0.040	3.225	0.159	0.121	**0.039**	3.103	0.155
Clustering	0.130	0.041	3.171	0.161	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Similarity	Cosine similarity											
Level	High interest				Medium interest				Low interest			
Metric	avg_inner	avg_rand	modularity	avg_center	avg_inner	avg_rand	modularity	avg_center	avg_inner	avg_rand	modularity	avg_center
TTD	0.736	**0.574**	**1.282**	0.857	0.573	**0.602**	0.952	0.761	0.475	0.629	0.755	0.695
LDA	**0.836**	0.660	1.267	0.917	**0.900**	0.612	**1.471**	**0.948**	**0.757**	**0.600**	**1.262**	**0.865**
SLPA	0.749	0.624	1.200	0.854	0.590	0.621	0.950	0.687	0.702	0.625	1.123	0.844
Clustering	0.763	0.622	1.226	0.875	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 9
Examples of user interests detected with TTD

user_10224	user_103043	user_113570
database (0.805), c#-dev (0.081)	java-dev (0.664), database (0.105)	c#-dev (0.393), web-dev (0.328)
sql-server (21)	java (135)	c# (107)
sql (21)	swing (28)	jquery (89)
tsql (6)	oracle (27)	javascript (56)
performance (4)	sql (23)	.net (47)
database (4)	subjective (15)	asp.net (27)
stored-procedures (3)	windows (13)	css (23)
sql-server-2005 (3)	eclipse (12)	regex (20)
.net (3)	best-practices (12)	html (20)
mysql (2)	plsql (10)	iphone (12)
sql-server-2000 (2)	regex (10)	string (10)
user_24181	**user_34509**	**user_30461**
web-dev (0.743), database (0.072)	c-dev (0.663), linux-dev (0.083)	ios-dev (0.885), linux-dev (0.020)
php (304)	c++ (703)	cocoa (333)
javascript (193)	c (187)	objective-c (184)
mysql (116)	templates (62)	iphone (47)
html (86)	stl (53)	cocoa-touch (39)
css (57)	linux (48)	osx (35)
regex (40)	subjective (45)	mac (34)
jquery (37)	pointers (44)	iphone-sdk (20)
sql (27)	java (42)	xcode (18)
ajax (26)	bash (40)	cocoa-bindings (18)
apache (23)	boost (31)	core-graphics (18)

quires a few hundreds of iterations to obtain stable topic distribution. Our model does not have this limitation.

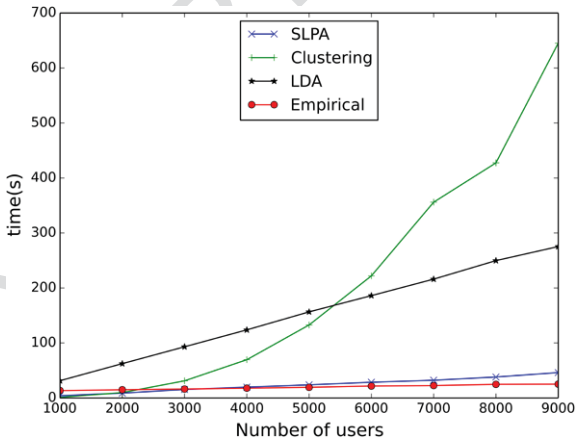

Fig. 8. Scalability of the compared user interest detection methods.

5.4. Discussion

To sum up, most community detection algorithms work well on real-life social networks which contain many *triangle-shape* structures. The interactions between the users in these networks are mainly based on their relationships. It is also noticeable that the relationships which a user in such network can maintain are limited and most likely restricted by the location (co-author networks in academia is also in this situation), so the overall structure of the network is *flatter*, *scattered* and with many *triangle-shape* structures. Comparatively, in Q&A sites, such as StackOverflow, there are no fixed relationships between users. Users interact with each other based on their own interests. And they are not aware of whom they are interacting with, so they will not maintain explicit relationships.

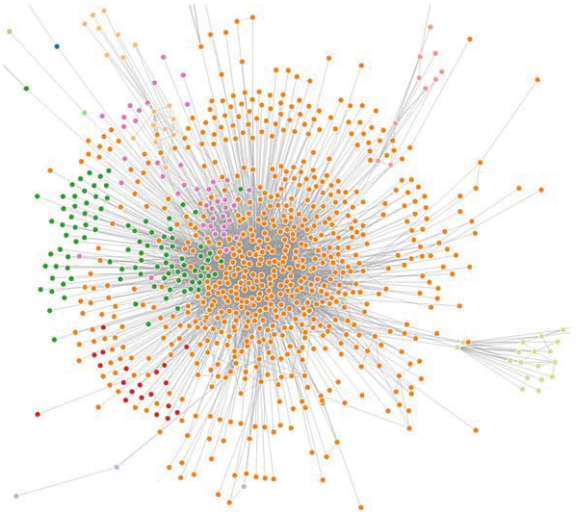

Fig. 9. Illustration of co-answer-network by SLPA [29].

Besides, a user can interact with any other user and mainly interacts with the "*gurus*" (most of questions are answered by a small group of people). So the overall structure of the network is *octopus-shape* [16] with less *triangle-shape* structures. According to [24], the average number of *triangle-shape* structures per user in Twitter dataset is around 35,714, while in our co-answer dataset, the number of *triangle-shape* structure per user is around 30 which is far less. So, graph-based community detection methods fail in such situation. The result of SLPA algorithm shows that it outputs one or two giant groups, together with many tiny groups that only contain a small number of users as depicted in Fig. 9, where each color represents a detected community. We can also see that the network contains less *triangle-shape* structures and a high-density *core*.It also indicates that the network has huge overlaps. Since clustering methods normally generate hard-partition communities, they cannot detect the overlapping communities which are typical in our case. Concerning the LDA-based methods, on one hand, in our dataset, question tag lists are quite short, and the experiment shows that our topic extraction method gives better results in this situation. On the other hand, the probabilistic graphical model requires hundreds of iterations to get stable results [9] which is more complicated and slower than our method. We also conducted similar experiments on a Flickr dataset in order to show that our method is not specific to StackOverflow. Recalling our research questions (How can we detect communities of interests in Q&A sites? How can we also identify the topics that attract them?) we believe we propose a topic detection method which is very suitable for Q&A datasets and an efficient user interest detection method to discover overlapping communities of interests.

Table 10
Output distributions of our model and their functionality

Notation	Functionality of distribution
θ_{uk}	detect a user's most interested topic
θ_{ku}	detect the most active users in a topic
θ_{kv}/θ_{kw}	detect the most relevant tags/words in a topic
θ_{kt}	detect the trends of a topic
θ_{tk}	detect the most popular topic at point in time
θ_{ukt}	detect a user's activity pattern in a topic
θ_{uke}	detect a user's most expertise topic

6. Temporal topic expertise activity (TTEA)

6.1. Problem definition

Let us consider StackOverflow for an example of the problem we address. In StackOverflow, as already explained, a user submits a question, then assigns between 1∼5 tags to indicate the key domains of the question. Other users who are interested in the question may provide answers to the question. Both questions and answers will get votes from other users. For instance, *Alice* posts a question and assigns it the tags {*html*, *css*, *height*}. Her question then gets *30* votes, and *Bob* gives an answer to this question at *10/11/2015*, that gets a voting score of *35*.

The Temporal Topic Expertise Activity (TTEA) model we propose aims at jointly modeling topics, topic trends, user expertise, and user activities. More precisely, we aim at extracting the information listed in Table 10.

6.2. Basic notions

Here are the basic notions later used in the description of TTEA:

Topic (θ_{kw}/θ_{kv}): A bag of words or tags which are closely related. Words are the content of questions or answers, tags are attached to questions. For example, the topic-tag distribution *Database*:{*mysql*: 0.5, *sql*: 0.3, *query*: 0.2}. expresses that topic *Database* is related to tags *mysql*, *sql*, and *query*.

User Topical Interest (θ_{uk}): A user is interested in different topics with different levels. For example, the user-topic distribution *Alice*:{*Database*: 0.8, *Java*:

0.2} expresses that *Alice* prefers to answer questions related to *Database*, but rather not about *Java*.

User Topical Activity (θ_{ku}): Different users are interested in the same topic with different levels. For example, the topic-user distribution *Database*:{*Alice*: 0.8, *Bob*: 0.2} expresses that *Alice* prefers to answer question related to *Database*, while *Bob* is not willing to contribute answers to it.

Topic Trend (θ_{kt}): A topic is popular at different points in time with different levels. For example,the topic-time distribution *Database*:{*May/2013*: 0.2, *June/2013*: 0.3, *July/2013*: 0.5} expresses that the topic *Database* is increasingly popular.

Topic Temporal Activity (θ_{tk}): Topics are active at a point in time with different levels. For example, the time-topic distribution *Sept/2013*:{*Ios*: 0.8, *Database*: 0.2} expresses that *ios* related questions are popular in Sept. 2013, while *Database* related questions are not specially popular.

User Topic Temporal Dynamics (θ_{ukt}): A user is interested in different topics at different points in time with different levels. For example, the topic-time distribution for *Alice ios*:{*May/2013*: 0.2, *June/2013*: 0.3, *July/2013*: 0.5} expresses that *Alice*'s interest to topic *ios* is increasing.

User Topical Expertise (θ_{uke}): A user has expertise in different topics with different levels. For example, the topic-expertise distribution for *Alice ios*:{*High*: 0.2, *Medium*: 0.7, *Low*: 0.1} expresses that *Alice*'s expertise on topic *ios* is probably in medium level.

6.3. TTEA model structure

TTEA is an LDA-based model. Figure 10 represents it using the plate notation.

Let $u_i \in \{1, 2, \ldots, U\}$ be the set of users, $p_i \in \{1, 2, \ldots, P\}$ the set of answer posts, which are generated by these users, $w_i \in \{1, 2, \ldots, W\}$ the set of words in answers posts, $ta_i \in \{1, 2, \ldots, Ta\}$ the set of tags which are attached to posts, $v_i \in \{1, 2, \ldots, V\}$ the set of votes for each answer posts, $ti_i \in \{1, 2, \ldots, Ti\}$ the set of points in time which could be months or days depending on the requirements, and $z_i \in \{1, 2, \ldots, K\}$ the set of topics for the posts. Here, U, P, W, Ta, V, Ti and K denote the total number of users, posts, words, tags, votes, points in time, and topics. α, β, δ, γ, η, and λ are Dirichlet priors. The notation and description of distributions θ_{uk}, θ_{kv}, θ_{kw}, θ_{kt}, and θ_{uke} are listed in Table 10.

Contrary to [4] who applied LDA model on long documents such as news articles and assumed that each word has a latent topic, we assume in TTEA that each answer post has one topic: like in other short social media, e.g. Twitter, an answer post is normally short, each answer post is therefore suitable to be assigned with one single latent topic, and all the words in that post are considered to be generated by this topic.

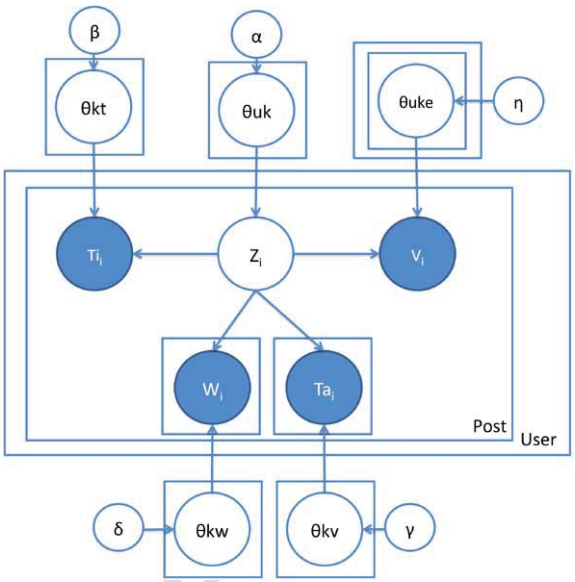

Fig. 10. TTEA model.

For expertise modeling, we do not use votes directly because (a) the vote scores are sparse and noncontinuous, and (b) it is not reasonable to tell that a vote score 55 is better than a vote score 50 if the vote score are ranging from 0 to 3,000. Since the vote scores' counts distribution follows a log distribution [32], we use the logarithmic value of vote score, and separate them into several expertise levels, which is one of the parameters: the expertise level.

For temporal modeling, like [12,27], we use time stamps directly. In order to model time at different levels, we simply split time stamps into different parts (month, day, and hour) and use them separately depending on the demands.

Let us consider a user u who wants to answer a question. She first selects a topic k according to her user-topic distribution θ_{uk}. Then she writes an answer post p. The words of p are generated from topic k's topic-word distribution θ_{kw}. Since only the questions have tags, we consider the answers automatically acquire all the tags of the question they respond to. Then the answer post p acquires its tags according to the topic-tag distribution θ_{kv} of topic k. Meanwhile, the

answer post p gets a time-stamp ti according to the topic-time distribution θ_{kt} of topic k. The generative process of TTEA model is described as follows.

- For the uth user, $u \in U$
  - draw user topic distribution $\theta_{uk} \sim \text{Dir}(\alpha)$
- For the kth topic, $k \in K$
  - draw topic tag distribution $\theta_{kv} \sim \text{Dir}(\gamma)$
  - draw topic word distribution $\theta_{kw} \sim \text{Dir}(\delta)$
  - draw topic time distribution $\theta_{kt} \sim \text{Dir}(\beta)$
- For the uth user, $u \in U$
  - for the kth topic, $k \in K$
    - draw user topic expertise distribution $\theta_{uke} \sim \text{Dir}(\eta)$
- For the uth user, $u \in U$
  - for the nth q&a post, $p \in P$
    - draw topic $z \sim \text{Multi}(\theta_{uk})$
    - draw time point $t \sim \text{Multi}(\theta_{kt})$
  - for the ith word, $w \in W$
    - draw word $w \sim \text{Multi}(\theta_{kw})$
  - for the jth tag, $ta \in Ta$
    - draw tag $t \sim \text{Multi}(\theta_{kv})$
    - draw expertise level $v \sim \text{Multi}(\theta_{uke})$

6.4. TTEA model inference

Like [12], we use the collapsed Gibbs sampling algorithm [9] to sample the hidden variable z, based on which the unknown probabilities $\{\theta_{uk}, \theta_{kv}, \theta_{kw}, \theta_{kt}$, and $\theta_{uke}\}$ can be estimated. For simplicity we set the hyper parameters to $\{\alpha, \beta, \delta, \gamma, \eta, \lambda\}$.

The TTEA inference process is as follows. We iteratively sample the topic indicator z_i for each answer post p_i according to equation (13). As explained before, each answer post will have one topic assignment.

$$
\begin{aligned}
&p(z_i = k | z_{\neg i}, \mathbf{U}, \mathbf{Ti}, \mathbf{Ta}, \mathbf{W}) \\
&\propto \frac{C^k_{u,\neg i} + \alpha_1}{\sum_{k=1}^{K} C^k_{u,\neg i} + K * \alpha_1} \\
&\cdot \frac{\prod_{ta=1}^{Ta} \prod_{q=0}^{C_{ta}-1} (C^{ta}_{k,\neg i} + q + \gamma)}{\prod_{p=0}^{\sum C_{ta}-1} \sum_{ta=1}^{Ta} (C^{v}_{k,\neg i} + p + Ta * \gamma)} \\
&\cdot \frac{\prod_{w=1}^{W} \prod_{s=0}^{C_w-1} (C^{w}_{k,\neg i} + s + \delta)}{\prod_{t=0}^{\sum C_w-1} \sum_{w=1}^{W} (C^{w}_{k,\neg i} + t + W * \delta)} \\
&\cdot \frac{C^{ti}_{k,\neg i} + \beta}{\sum_{ti=1}^{Ti} C^{ti}_{k,\neg i} + Ti * \beta} \\
&\cdot \frac{C^{e}_{u,k,\neg i} + \eta}{\sum_{e=1}^{E} C^{e}_{u,k,\neg i} + E * \eta}
\end{aligned}
\tag{13}
$$

where $\neg i$ enforces that all the counters used are calculated with the answer post p_i excluded. $C^k_{u,\neg i}$ is the number of posts by user u assigned to topic k, C_{ta} is the number of tags ta in p_i, therefore, $\sum C_{ta}$ is the total number of tags in p_i, $C^{ta}_{k,\neg i}$ is the number of tags ta assigned to topic k. Similarly, C_w is the number of words w in p_i, $\sum C_w$ is the number of words in p_i, $C^w_{k,\neg i}$ is the number of words w assigned to topic k. $C^{ti}_{k,\neg i}$ is the number of posts assigned to topic k and posted at time ti. $C^e_{u,k,\neg i}$ is the number of posts which are assigned to topic k and got a vote score in the range of expertise level e.

Then, with the result of the Gibbs sampling algorithm, we can make the following parameter estimation:

$$\theta_{uk} = \frac{C^k_u + \alpha_1}{\sum_{k=1}^{K} C^k_u + K * \alpha_1} \tag{14}$$

$$\theta_{kv} = \frac{C^{ta}_k + \gamma}{\sum_{ta=1}^{Ta} C^{ta}_k + Ta * \gamma} \tag{15}$$

$$\theta_{kw} = \frac{C^w_k + \delta}{\sum_{w=1}^{W} C^w_k + W * \delta} \tag{16}$$

$$\theta_{kt} = \frac{C^{ti}_k + \beta_2}{\sum_{ti=1}^{Ti} C^{ti}_k + Ti * \beta_2} \tag{17}$$

$$\theta_{uke} = \frac{C^e_{u,k} + \eta}{\sum_{e=1}^{E} C^e_{u,k} + E * \eta} \tag{18}$$

6.5. Post processing

The above model can only generate the distributions $\{\theta_{uk}, \theta_{kv}, \theta_{kw}, \theta_{kt}$, and $\theta_{uke}\}$. To generate the other distributions, e.g. θ_{ku}, θ_{tk} and θ_{ukt}, we directly use the sample results at each iteration and keep recording the corresponding counters. Therefore, C^u_k is the number of posts assigned to topic k and posted by user u, C^k_{ti} is the number of posts posted at time ti and assigned to topic k. $C^{ti}_{u,k}$ is the number of posts by user u, assigned

Table 11
Basic statistics on the dataset

number of tags	32,379
number of questions	4,592,961
number of users asking questions	833,041
number of users providing answers	8,585,113
number of questions having accepted answers	2,808,825

to topic k and posted at time ti. Then, we estimate θ_{ku}, θ_{tk}, θ_{ukt} according to the following equations:

$$\theta_{ku} = \frac{C_k^u + \alpha_2}{\sum_{u=1}^{U} C_k^u + U * \alpha_2} \tag{19}$$

$$\theta_{tk} = \frac{C_{ti}^k + \beta_1}{\sum_{k=1}^{K} C_{ti}^k + K * \beta_1} \tag{20}$$

$$\theta_{ukt} = \frac{C_{u,k}^{ti} + \lambda}{\sum_{ti=1}^{T} C_{u,k}^{ti} + T * \lambda} \tag{21}$$

7. TTEA experiments and evaluation on StackOverflow data

7.1. Dataset description

We conducted experiments on a dataset from StackOverflow. This site releases its whole content every three month. For our experiments, we used the data dump from July 2008 to March 2013. Table 11 and Fig. 11 provide basic statistics on the dataset.

Here are some general observations about the dataset: (1) nearly half of the questions do not have accepted answers; (2) nearly half of the questions only have one answer and it maybe inadequate; (3) more than a third of the questions only have one or two tags; (4) nearly half of the users only answer one question so question routing and incentives are important problems; (5) nearly 10% percent of the questions do not have answers.

Due to the large volume of the dataset over 3 years, the processing time is extremely long. To simplify the processing, for the following experiments, we randomly chose several continuous months from the dataset, with no bias to the selections.

7.2. Compared methods

To evaluate the effectiveness of our model, we compared it with several related works:

- TTEA is our method for modeling user, topic, temporal and expertise in Q&A sites. Besides, we also model activities by adding virtual nodes. We can generate the user-topic distribution and topic-activity distribution simultaneously.
- TEM: [32] proposed a model for user, topic and expertise in Q&A sites. It integrates a Gaussian Mixture Model to model expertise, which is time consuming. We simplify this process by directly modeling votes information. Besides, it does not model temporal information and user topic activities.
- UQA: [10] proposed a User–Question–Answer model for modeling users and topics in Q&A sites. In certain Q&A sites, questions have category information which have proved to be very useful. The category in their model is similar to tags in TTEA model and TEM model. However we allow multiple tags for each posts while they can only set a single category.
- GrosToT: [12] proposed a User–Group–Topic–Time model for modeling users, groups, topics and time in social media sites. It introduces a group level between user and topic compared with other models. It does not directly generate user-topic distribution, so we compute it with the user-group distribution and group-topic distribution.
- LDA: based on [4] we apply LDA model to create a User–Topic–Post model for modeling users and topics. It can generate the user-topic distribution and topic-words distribution.

We choose the same number of topics K = 30 as [7] and the same number of expertises E = 10 as [32], which have proved to be a reasonable setting for the Stackoverflow dataset. We empirical set Dirichlet hyper parameters $\alpha_1 = \alpha_2 = 50/K$, $\beta_1 = \beta_2 = 0.01$, $\delta = \lambda = \eta = 0.01$, $\gamma = 0.001$ according to suggestions in [9].

7.3. Performance of topic extraction

Table 12 and Table 13 show the top tags and words detected by our model. We use again the Perplexity [4] metric as a quantitative way to measure the performance of topic extraction.

We include in our training dataset all the posts in the two months from August 1st 2011 to October 1st 2011, from users having more than 80 posts (as in [32]). The resulting training dataset contains 87,516 q&a posts by

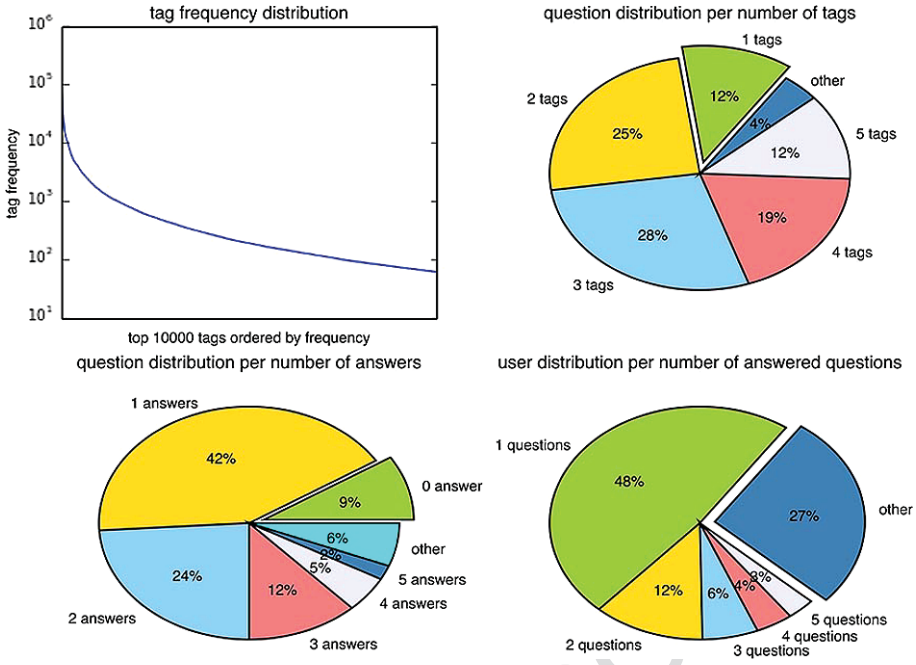

Fig. 11. Basic perspectives of the dataset.

Table 12
Top tags for different topics generated by the TTEA model

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
php	c#	iphone	c++	javascript	android	sql	java	jquery	git
xslt	.net	objective-c	c	jquery	java	mysql	spring	javascript	svn
xml	linq	ios	pointers	php	android-layout	sql-server	eclipse	html	version-control
xpath	generics	xcode	templates	ajax	listview	php	jsp	css	github
mysql	asp.net	cocoa-touch	stl	html	activity	query	.htaccess	jquery-selectors	mercurial
html	vb.net	ipad	arrays	json	android-intent	tsql	servlets	jquery-ui	eclipse
arrays	c#-4.0	uitableview	vector	asp.net	sqlite	sql-server-2008	jsf	dom	tortoisesvn
jquery	reflection	iphone-sdk-4.0	string	jquery-ajax	layout	join	mod-rewrite	php	linux
javascript	entity-framework	cocoa	function	forms	android-widget	select	maven	javascript-events	clearcase
foreach	list	xcode4	c++11	asp.net-mvc-3	xml	sql-server-2005	apache	ajax	ssh

674 users. For data preprocessing, we tokenize text and removed the stop words. For the testing dataset, we use all the posts of the same set of users than the training data but this time from October 1th 2011 to January 1th 2012. So training and testing datasets have no overlap but concern the same community. We vary the number of topics: 10, 30, 50, and 100. For a testing set of M posts, N_i denotes the number of words in the ith post and the Perplexity score is computed according to equation (22).

$$\text{Perplexity}(D_{\text{test}}) = \exp\left\{-\frac{\sum_{i=1}^{M}\log p(W_i)}{\sum_{i=1}^{M} N_i}\right\} \tag{22}$$

Table 13
Top words for different topics generated by the TTEA model

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
xsl	aspx	view	std	jquery	android	select	html	jquery	git
td	msdn	reference	const	ajax	activity	join	java	div	branch
tr	microsoft	nsstring	pointer	script	html	group	file	click	commit
template	library	apple	char	javascript	view	order	spring	element	file
select	select	html	template	page	developer	table	jar	event	svn
row	linq	library	vector	html	intent	key	apache	input	repo
echo	system	documentation	operator	form	reference	count	eclipse	document	repository
table	dictionary	developer	compiler	url	layout	row	docs	text	files
match	ienumerable	ios	memory	document	try	inner	servlet	html	master
node	expression	release	struct	json	button	query	web	api	github

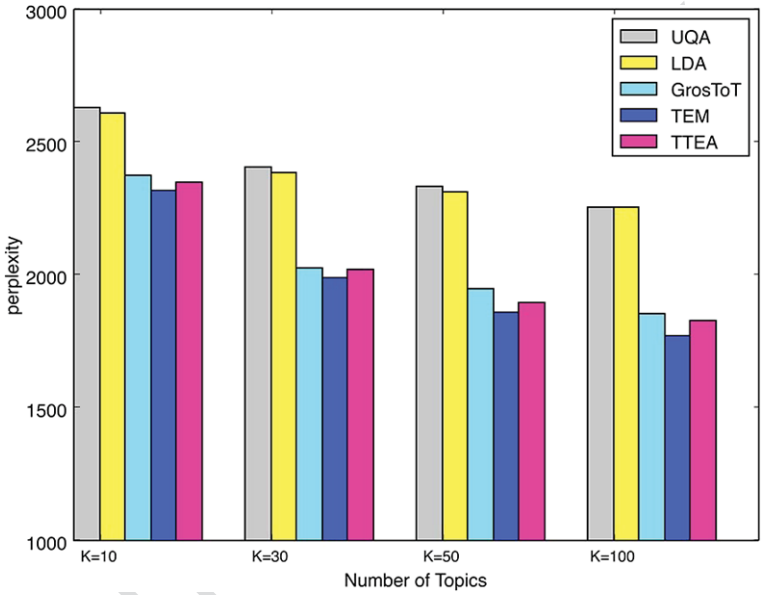

Fig. 12. Comparison of topic extraction performances.

where $p(W_i)$ is the probability of the words in the test document d_i. In our model, $p(W_i)$ is computed according to equation (23).

$$P(W_i) = \sum_k \theta_{u_i k} \prod_w \theta_{k w_i} \tag{23}$$

Figure 12 shows the perplexity results for our TTEA method and other state-of-the-art methods. TTEA is almost as good as TEM. But TEM integrates a Gaussian Mixture Model, which is time consuming. The training process of TEM is nearly three times longer than the other models.

[6] suggested that topic models should focus on evaluations on real-world task performance rather than on optimizing likelihood-based measures. So, in addition to the perplexity-based evaluation, we used the results of TTEA to perform real-word tasks and we evaluated them. This is described in the following subsections.

7.4. Question routing

Given a question q and a set of users U, the task is to rank all these users by their interests to answer question q. We score each user u by considering the similarity between his topics of interest and the topics of the question ($\mathrm{Sim}(u, q)$). The intuition behind equation (24) is that the more a user is interested in the topic of a question, the more likely he is to provide an

answer to that question.

$$\mathrm{Sim}(u, q) = \big(1 - \mathrm{JS}(\theta_{uk}, \theta_{qk})\big) \tag{24}$$

where θ_{uk} is the user topic interest distribution, θ_{qk} is the question topic distribution, and JS$(\cdot)$ is the Jensen–Shannon divergence distance. We obtain θ_{uk} directly from model results. For θ_{qk}, we apply equation (25).

$$\begin{aligned} \theta q, k &\propto p(k|w_q, t_q, u) \\ &= p(k|u)p(w_q|k)p(t_q|k) \\ &= \theta uk \sum_{w_i \in w_q} \theta_{kw_i} \sum_{t_i \in t_q} \theta_{kv_i} \end{aligned} \tag{25}$$

where w_q and t_q are the sets of all the words and tags in question q and θkw, θkv are the topic-word distribution and topic-tag distribution obtained directly from the model result. Then for question q, we compute the Sim score for user set U and rank them in decreasing order.

We used all the posts from July 1th 2011 to October 1th 2011 from users having more than 50 q&a posts for the training dataset. Rather than using the threshold of 80 post like in [32], we empirically set it to 50 posts to get enough users for recommendation. The resulting training set contains 297,881 posts by 2,555 users. For the testing dataset, we use all the questions posted by the same set of users as in the training set but this time from October 1th 2011 to January 1th 2012. Therefore the training and testing datasets have no overlaps. We removed testing questions which have no, or only one, answer. The resulting test dataset contains 6,044 questions, 18,077 answers and 7,888 involved users.

We also chose another period for this experiment. Besides, we vary the number of topics by 15 and 50, we vary the filter limit by 40 and 80. These experiment results are shown in Section 7.5.

In order to evaluate different models, we consider precision at position N (Precision@N or simply P@N) and recall at position N (Recal@N or simply R@N), which are widely used measures in the Information Retrieval community. Let R_q be the recommendations of users for a question q and U_q be the actual set of users who posted for question q. Then Precision@N is defined in equation (26) and Recal@N is defined in equation (27).

$$\mathrm{P@N} = \frac{1}{|Q|} \sum_{q \in Q} \frac{|R_q \cap U_q|}{|R_q|} \tag{26}$$

$$\mathrm{R@N} = \frac{1}{|Q|} \sum_{q \in Q} \frac{|R_q \cap U_q|}{|U_q|} \tag{27}$$

where Q is the set of testing questions. Like in [7], we use the Matching Set Count (MSC) which is defined in equation (28). The idea is to count the number of successful recommendations, i.e., for which at least one of the recommended users answered the question.

$$\mathrm{MSC@N} = \frac{1}{|Q|} \sum_{q \in Q} 1[R_q \cap U_q \neq \emptyset] \tag{28}$$

where 1[condition] is equal to 1 if condition is true, otherwise 0.

In addition, our model can capture activity and we believe this information improves question routing. The intuition is that even if a user has a high Sim score for a question, the less he is active, the less likely he is to provide an answer to that question. Therefore, we define a score SimAct to combine both topic similarity and activity level as shown in equation (29), where $\mathrm{Act}(u, q)$ is the computed activity score for user u to question q. A high value of the Act score indicates a high probability of activity on a question. We use TTEA to denote the method using only the similarity information, that is to say, ranking users by Sim score. We use TTEA-ACT to denote the method using both similarity and activity, that is to say, ranking users by SimAct score. We also integrated our activity model to the TEM model and we refer to it as TEM-ACT.

$$\begin{aligned} \mathrm{SimAct}(u, q) &= \big(1 - \mathrm{JS}(\theta_{uk}, \theta_{qk})\big) * \mathrm{Act}(u, q) \\ &= \big(1 - \mathrm{JS}(\theta_{uk}, \theta_{qk})\big) \\ &\quad * \sum_{k=1}^{K} \theta_{qk} * \theta_{ku} \end{aligned} \tag{29}$$

Table 14 shows the results. We ran the experiments five times and listed the average scores. Our observations can be summarized as follows: (1) UQA and GROSTOT perform the better when the number of recommended users are small, and TTEA and TEM begin to outperform UQA and GROSTOT when the number of recommended users is large; (2) TTEA-ACT shows the best performances compared with the baseline competitors; (3) both TTEA-ACT and TEM-ACT perform better than the other models. The activity modeling is a generic method that could improve the performance not only of our model, but also of other models although here we only show the result for the activity model with TEM as an example; (4) even

Table 14
Question Routing experiments, Random denotes that we randomly recommend users for the test questions

	p@5	p@10	p@20	p@30	r@5	r@10	r@20	r@30	msc@5	msc@10	msc@20	msc@30
TTEA	0.024	0.019	0.015	0.013	0.045	0.072	0.111	0.142	0.112	0.178	0.269	0.339
TTEA-ACT	0.028	**0.022**	**0.017**	**0.014**	0.052	**0.083**	**0.127**	**0.159**	0.134	**0.209**	**0.313**	**0.382**
TEM	0.024	0.019	0.015	0.013	0.045	0.073	0.114	0.146	0.114	0.179	0.275	0.344
TEM-ACT	0.029	**0.023**	**0.018**	**0.015**	0.054	**0.084**	**0.129**	**0.162**	0.137	**0.210**	**0.315**	**0.388**
UQA	**0.030**	0.019	0.012	0.010	**0.062**	0.075	0.095	0.112	**0.149**	0.179	0.224	0.261
GROSTOT	0.027	0.017	0.011	0.009	0.055	0.067	0.085	0.099	0.134	0.164	0.204	0.236
RANDOM	0.001	0.001	0.001	0.001	0.001	0.002	0.005	0.007	0.003	0.007	0.013	0.019

Table 15
Question Routing Experiments on Another Dataset

	p@5	p@10	p@20	p@30	r@5	r@10	r@20	r@30	msc@5	msc@10	msc@20	msc@30
TTEA	0.026	0.020	0.015	0.013	0.047	0.073	0.110	0.136	0.123	0.186	0.273	0.332
TTEA-ACT	**0.032**	**0.026**	**0.019**	**0.016**	**0.058**	**0.093**	**0.137**	**0.168**	**0.153**	**0.236**	**0.339**	**0.405**
TEM	0.025	0.021	0.016	0.013	0.047	0.076	0.112	0.139	0.120	0.191	0.274	0.333
TEM-ACT	**0.032**	**0.025**	**0.020**	**0.016**	**0.058**	**0.092**	**0.141**	**0.171**	**0.153**	**0.235**	**0.348**	**0.411**
UQA	0.027	0.016	0.011	0.009	0.052	0.062	0.080	0.096	0.130	0.155	0.196	0.233
GROSTOT	0.023	0.014	0.009	0.007	0.044	0.055	0.069	0.081	0.112	0.137	0.172	0.200
RANDOM	0.001	0.001	0.001	0.001	0.001	0.002	0.004	0.005	0.003	0.005	0.010	0.015

Table 16
Question Routing experiments with 15 topics

	p@5	p@10	p@20	p@30	r@5	r@10	r@20	r@30	msc@5	msc@10	msc@20	msc@30
TTEA	0.016	0.013	0.012	0.010	0.030	0.050	0.086	0.112	0.076	0.127	0.213	0.269
TTEA-ACT	0.023	**0.018**	**0.015**	**0.012**	0.042	**0.066**	**0.107**	**0.134**	**0.112**	**0.170**	**0.268**	**0.329**
TEM	0.017	0.015	0.012	0.010	0.032	0.054	0.091	0.115	0.083	0.137	0.222	0.276
TEM-ACT	0.024	**0.018**	**0.014**	**0.012**	0.043	**0.068**	**0.103**	**0.131**	**0.114**	**0.172**	**0.254**	**0.319**
UQA	**0.028**	0.016	0.011	0.008	**0.056**	**0.066**	0.083	0.099	0.137	0.159	0.199	0.238
Grostot	0.023	0.015	0.010	0.008	0.045	0.058	0.075	0.089	0.112	0.143	0.183	0.216
Random	0.001	0.001	0.001	0.001	0.002	0.003	0.004	0.006	0.005	0.008	0.012	0.017

if TEM or TEM-ACT perform better than our model they remain again time consuming. Experiments show that the training process takes around 3∼4 times longer compared to our model.

7.5. Experiment parameter sensitivity analysis

For the training dataset, we used all the posts in a three months period, from January 1th 2011 to March 31th 2011, from users having at least 50 q&a posts, rather than 80 posts like [32], in order to get enough users for recommendations. The training set contains 371,181 posts by 3,123 users. For the testing dataset, we used all the questions posted by the same set of users as in the training set, but this time from April 1th 2011 to June 31th 2011. Therefore the training and testing datasets have no overlaps. We removed questions with no or only one answer. The resulting test dataset contains 9,048 questions, 27,870 answers and 10,147 users. Table 15 shows the question routing results. We can still find that TTEA-ACT outperforms all the baseline models. Besides, Both TTEA-ACT and TEM-ACT outperform all the other models.

Table 16 shows the question routing results with a number of topics set to 15. We use the same training and testing datasets as in Section 7.4.

Table 17 shows the question routing results for the number of topics set to 50. We use the same training and testing datasets as in Section 7.4.

Table 18 shows the question routing results with users having more than 40 posts. We use the same period of dataset used in Section 7.4. Due to the different filter limit, the training set contains 3,457 users and

Table 17
Question Routing experiments with 50 topics

	p@5	p@10	p@20	p@30	r@5	r@10	r@20	r@30	msc@5	msc@10	msc@20	msc@30
TTEA	0.028	0.023	0.018	0.015	0.054	0.087	0.132	0.168	0.134	0.215	0.319	0.394
TTEA-ACT	**0.033**	**0.025**	**0.019**	**0.016**	0.063	**0.095**	**0.142**	**0.178**	**0.158**	**0.235**	**0.343**	**0.418**
TEM	0.029	0.024	0.018	0.015	0.056	0.088	0.136	0.171	0.141	0.220	0.325	0.400
TEM-ACT	**0.033**	**0.026**	**0.020**	**0.017**	0.062	**0.096**	**0.145**	**0.182**	**0.157**	**0.240**	**0.347**	**0.427**
UQA	0.032	0.019	0.012	0.010	**0.065**	0.077	0.097	0.116	**0.158**	0.185	0.227	0.270
Grostot	0.028	0.017	0.011	0.009	0.056	0.067	0.088	0.102	0.136	0.163	0.210	0.241
Random	0.001	0.001	0.001	0.001	0.002	0.002	0.005	0.007	0.004	0.006	0.013	0.018

Table 18
Question Routing experiments, with users having more than 40 posts

	p@5	p@10	p@20	p@30	r@5	r@10	r@20	r@30	msc@5	msc@10	msc@20	msc@30
TTEA	0.021	0.018	0.014	0.012	0.040	0.067	0.104	0.132	0.100	0.167	0.253	0.313
TTEA-ACT	0.026	**0.021**	**0.016**	**0.014**	0.049	**0.076**	**0.118**	**0.149**	0.126	**0.193**	**0.292**	**0.360**
TEM	0.023	0.018	0.014	0.012	0.043	0.069	0.106	0.137	0.109	0.170	0.255	0.323
TEM-ACT	0.027	**0.021**	**0.016**	**0.014**	0.050	**0.078**	**0.121**	**0.152**	0.128	**0.194**	**0.295**	**0.362**
UQA	**0.029**	0.018	0.011	0.009	**0.059**	0.071	0.087	0.101	**0.142**	0.169	0.205	0.235
Grostot	0.025	0.016	0.010	0.008	0.050	0.063	0.077	0.091	0.122	0.152	0.188	0.217
Random	0.000	0.000	0.000	0.000	0.001	0.002	0.003	0.005	0.002	0.004	0.008	0.013

Table 19
Question Routing experiments, with users having more than 80 posts

	p@5	p@10	p@20	p@30	r@5	r@10	r@20	r@30	msc@5	msc@10	msc@20	msc@30
TTEA	0.028	0.023	0.019	0.016	0.051	0.083	0.135	0.175	0.132	0.212	0.336	0.424
TTEA-ACT	0.031	**0.026**	**0.020**	**0.018**	0.058	**0.094**	**0.146**	**0.188**	0.150	**0.238**	**0.364**	**0.457**
TEM	0.031	0.026	0.020	0.017	0.056	0.095	0.147	0.188	0.143	0.238	0.356	0.445
TEM-ACT	0.035	**0.027**	**0.021**	**0.018**	0.063	**0.100**	**0.151**	**0.193**	0.165	**0.253**	**0.375**	**0.468**
UQA	**0.040**	0.025	0.016	0.013	**0.077**	0.096	0.124	0.150	**0.194**	0.237	0.299	0.357
Grostot	0.036	0.022	0.015	0.012	0.070	0.086	0.114	0.135	0.177	0.214	0.278	0.325
Random	0.001	0.001	0.001	0.001	0.002	0.003	0.006	0.011	0.005	0.008	0.019	0.030

338,485 q&a posts, the testing set contains 8,579 questions, 25,500 answers and 10,135 involved users.

Table 19 shows the question routing results with users having more than 80 posts. We use the same period of dataset used in Section 7.4. Due to the different filter limit, the training set contains 1,275 users and 216,940 q&a posts, the testing set contains 2,589 questions, 8,006 answers and 4,196 involved users.

7.6. Recommendation of expert users

Given a question q and a set of users U, the task is now to recommend N users until one of the users gets the highest vote. The point is to rank recommended users by their expertise to answer question q. We score each user u by considering the similarity $\mathrm{SimExp}(u, q)$ between user topic interest and user topic expertise to answer question q. The intuition behind equation (30) is that if the user is interested in the question, she will probably provide an answer to that question and if the user has expertise on the question, the answer will probably have the highest vote score.

$$\mathrm{SimExp}(u, q) = \big(1 - \mathrm{JS}(\theta_{uk}, \theta_{qk})\big) * \mathrm{Exp}(u, q) \tag{30}$$

where θ_{uk}, θ_{qk} is the same than in (24) for user topic interest distribution. For our method, we compute $\mathrm{Exp}(u, q)$ by equation (31)

$$\mathrm{Exp}(u, q) = \sum_{e=1}^{E} \theta_{kue} * e \tag{31}$$

Table 20
Expert recommendation experiments

Methods	N = 30	N = 60	N = 100
TEM	0.128	**0.228**	0.392
TTEA	0.079	0.195	**0.443**
UQA	**0.146**	0.206	0.261
Grostt	0.127	0.172	0.220
Random	0.008	0.018	0.028

As UQA and GROSTOT do not model expertise, like [32], we set $\text{Exp}(u, q)$ to 1 for these two methods. For TEM, we reuse equation (32) indicated in [32].

$$\text{Exp}(u, q) = \sum_{e=1}^{E} \phi_{z,u,e} * \mu_e \tag{32}$$

In order to evaluate different models, we consider the percentage of successful expert recommendation until position N. A successful expert recommendation until position N means that the Nth user, recommended by an algorithm, not only answers the question but also gets the highest votes.

Table 20 shows the results. Random denotes that we randomly recommend users for the test questions. We ran the experiments five times and listed the average scores. We summarize our observations as follows: (1) Our TTEA shows the best performances compared with the baseline models when the number of recommended users is large. This means that when we recommend 100 users for each testing questions, in around 44% cases we have one user not only answering the question, but also winning the highest vote. (2) When the number of recommended users is large, both TEM and TTEA perform better than other models which do not model expertise, so expertise modeling can improve expert recommendation. (3) TEM uses Gaussian Mixture Model to model expertise, while we directly model votes which is less precise. Therefore, we perform badly when the number of recommended users is small. (4) After ranking users by topic similarity scores, using expertise scores to re-rank those users actually lowers the probability of the top ranked user to answer the question. The intuition behind is that a user having high expertise on a question does not necessarily have high topic similarity score with the question.

7.7. Trends

With the temporal modeling of TTEA, we can explore topic dynamics at many different levels. We present illustrative case studies to show the advantage of temporal modeling.

We first set the time window at the month level. Figure 13(a) shows the dynamics of *Android*, *Iphone* and *Flash* related topics at different months from Jan 2011 to Dec 2011. *Flash* related topics are more active in the early of 2011, but become less popular in the late of 2011. We then set the time window at the day level. Figure 13(b) shows the dynamics of *Android*, *Iphone* and *Flash* related topics from July 1st 2011 to July 31st 2011. We can see that all topics are active from Monday to Friday, and not active during the weekend. Lastly, we set the time window at the hour level. Figure 13(c) shows the dynamics of *Android*, *Iphone* and *Flash* related topics at different hours during a day. We can verify that both *Android* and *Iphone* related topics are more active during daytime, but *Flash* related topics are more active during the afternoon.

Previous figures show the topic dynamics on a global level. We now illustrate the topic dynamics at the user level. We choose top active users according to the output of θ_{ku} in *Android* related topic and *Iphone* related topic separately. Figure 14(a), (b) show the activity pattern of the two most active users in *Iphone* related topic. We can observe that the user in Fig. 14(a) is only active during work-time. The user seldom answers questions after 7PM. On the contrary, the user in Fig. 14(b) is active until very late but not midnight. Figure 14(c), (d) show the activity pattern of the two most active users in *Android* related topic. We can observe that the user in Fig. 14(c) is active in the morning, afternoon and evening. On the contrary, the user in Fig. 14(d) is even active at midnight. For all these users, we can observe that they are not actually active on the topics they are not interested in. We believe this information will benefit many community management related tasks.

8. Conclusion and future work

In this work, we addressed three research questions: *How can we identify the common topics binding user together? How can we detect topic based overlapping communities? How can we extract topic based expertise and temporal dynamics?* By applying the original LDA model on these tasks, we encountered three problems. The first one is a lack of efficiency: the complexity of the probabilistic model was prohibitive. The second problem is that the original LDA model is not enough to extract temporal and expertise information.

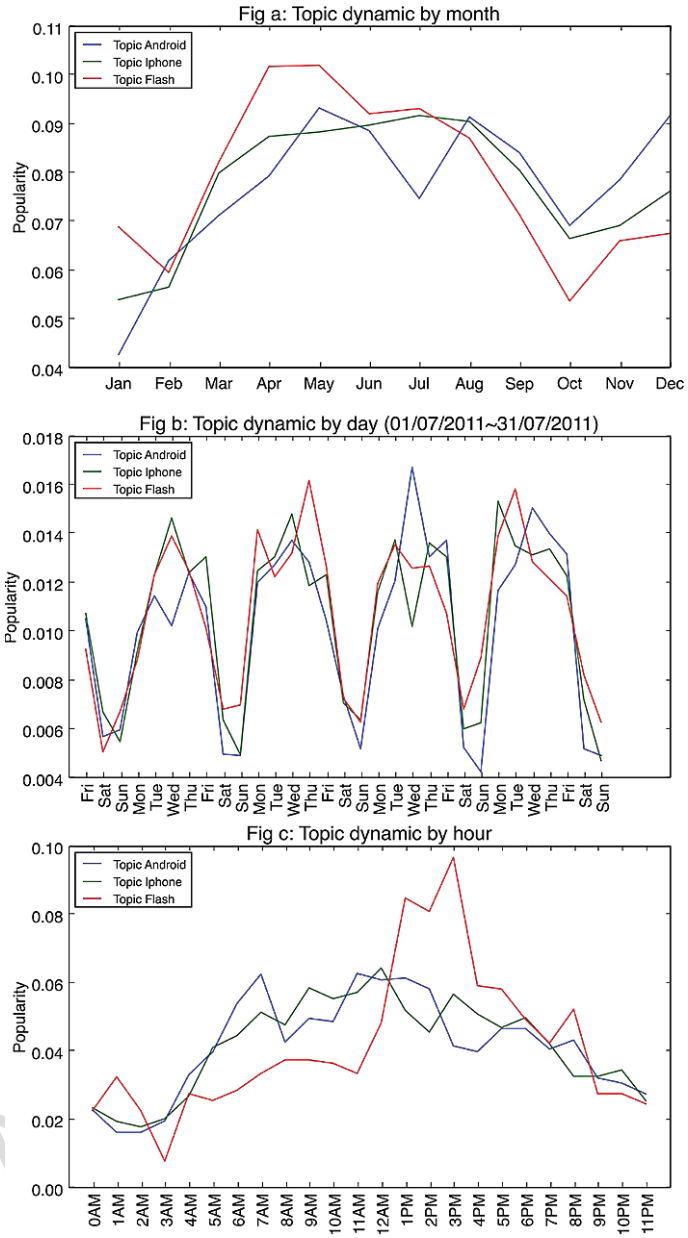

Fig. 13. Topic dynamics.

The third one is an incomparability problem. The detected probabilities distributions cannot be compared with each other. Therefore, firstly, we proposed TTD a simpler method to detect topics and overlapping communities to solve the first problem. We conducted experiments on a dataset from the popular Q&A site StackOverflow to compare different approaches. The results indicate that for this kind of web communities our method can be a good replacement to more complicated methods for detecting overlapping communities of interests. Secondly, we proposed TTEA a more complex model to extract more information from user generated content and to fix the others problems. Our model can simultaneously uncover the topics, activities, expertise and temporal dynamics. This extracted information can enable us to improve tasks such as: question routing, expert recommendation and community life-cycle management. Again, we conducted ex-

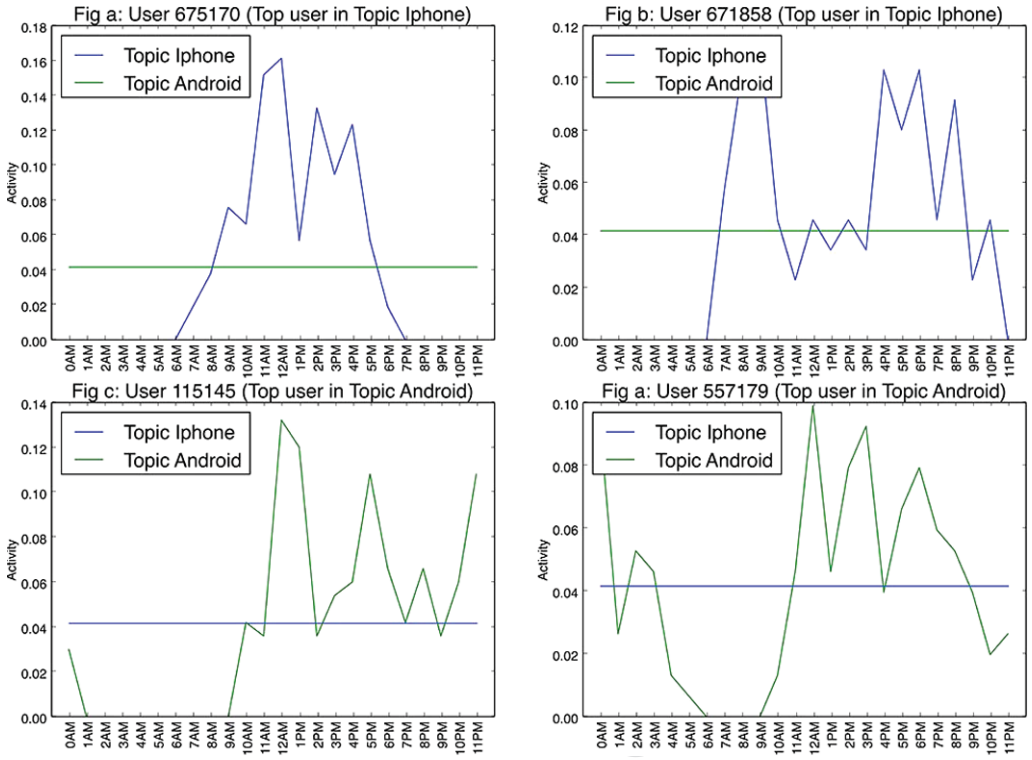

Fig. 14. User topic activities.

periments on StackOverflow dataset. We demonstrated that TTEA shows advantages in topic modeling. It also achieves good performances on question routing task and expert detection task compared with the state of the art models. We also illustrated that our model can detect user and topic temporal dynamics which could be used on user life-cycle management.

There are many future directions for this work. It is obvious that the proposed models and methods are not limited to the processing of Q&A datasets and we intend to adapt them to other kinds of social media.

Acknowledgements

The authors would like to thank StackOverflow for sharing their data. We thank the ANR-12-CORD-0026 Ocktopus project grant for the support of this research.

References

[1] L.A. Adamic and B.A. Huberman, Power-law distribution of the world wide web, *Science* **287**(5461) (2000), 2115. doi:10.1126/science.287.5461.2115a.

[2] Y.-Y. Ahn, J.P. Bagrow and S. Lehmann, Link communities reveal multiscale complexity in networks, *Nature* **466**(7307) (2010), 761–764. doi:10.1038/nature09182.

[3] A. Anderson, D. Huttenlocher, J. Kleinberg and J. Leskovec, Discovering value from community activity on focused question answering sites: A case study of stack overflow, in: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2012, pp. 850–858. doi:10.1145/2339530.2339665.

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

[5] M. Bouguessa, B. Dumoulin and S. Wang, Identifying authoritative actors in question-answering forums: The case of Yahoo! answers, in: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2008, pp. 866–874. doi:10.1145/1401890.1401994.

[6] J. Chang, J. Boyd-Graber, C. Wang, S. Gerrish and D.M. Blei, Reading tea leaves: How humans interpret topic models, in: *Neural Information Processing Systems*, 2009.

[7] S. Chang and A. Pal, Routing questions for collaborative answering in community question answering, in: *Proceedings of the 2013 IEEE/ACM ASONAM*, ACM, New York, NY, USA, 2013, pp. 494–501.

[8] U. Gargi, W. Lu, V.S. Mirrokni and S. Yoon, Large-scale community detection on Youtube for topic discovery and exploration, in: *ICWSM*, 2011.

[9] T.L. Griffiths and M. Steyvers, Finding scientific topics, *Proceedings of the National Academy of Sciences* **101**(suppl. 1) (2004), 5228–5235. doi:10.1073/pnas.0307752101.

[10] J. Guo, S. Xu, S. Bao and Y. Yu, Tapping on the potential of q&a community by recommending answer providers, in: *Proceedings of the 17th ACM CIKM*, ACM, 2008, pp. 921–930.

[11] T. Hofmann, Probabilistic latent semantic analysis, in: *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers Inc., 1999, pp. 289–296.

[12] Z. Hu, J. Yao and B. Cui, User group oriented temporal dynamics exploration, in: *Twenty-Eighth AAAI14*, 2014.

[13] Z. Ji and B. Wang, Learning to rank for question routing in community question answering, in: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, ACM, 2013, pp. 2363–2368. doi:10.1145/2505515.2505670.

[14] P. Jurczyk and E. Agichtein, Discovering authorities in question answer communities by using link analysis, in: *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, ACM, 2007, pp. 919–922. doi:10.1145/1321440.1321575.

[15] A. Lancichinetti, F. Radicchi, J.J. Ramasco and S. Fortunato, Finding statistically significant communities in networks, *PLoS One* **6**(4) (2011), e18961. doi:10.1371/journal.pone.0018961.

[16] J. Leskovec, K.J. Lang, A. Dasgupta and M.W. Mahoney, Statistical properties of community structure in large social and information networks, in: *Proceedings of the 17th International Conference on World Wide Web*, ACM, 2008, pp. 695–704.

[17] B. Li and I. King, Routing questions to appropriate answerers in community question answering services, in: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ACM, 2010, pp. 1585–1588.

[18] D. Li, B. He, Y. Ding, J. Tang, C. Sugimoto, Z. Qin, E. Yan, J. Li and T. Dong, Community-based topic modeling for social tagging, in: *Proc. of the 19th ACM CIKM, CIKM '10*, ACM, New York, NY, USA, 2010, pp. 1565–1568.

[19] Z. Ma, A. Sun, Q. Yuan and G. Cong, A tri-role topic model for domain-specific question answering, in: *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[20] A. McDaid and N. Hurley, Detecting highly overlapping communities with model-based overlapping seed expansion, in: *2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, IEEE, 2010, pp. 112–119. doi:10.1109/ASONAM.2010.77.

[21] P. Mika, Ontologies are us: A unified model of social networks and semantics, *Web Semantics: Science, Services and Agents on the World Wide Web* **5**(1) (2007), 5–15. doi:10.1007/978-3-540-76298-0.

[22] A.Y. Ng, M.I. Jordan and Y. Weiss, On spectral clustering: Analysis and an algorithm, in: *Advances in Neural Information Processing Systems*, MIT Press, 2001, pp. 849–856.

[23] A. Pal, R. Farzan, J.A. Konstan and R.E. Kraut, Early detection of potential experts in question answering communities, in: *User Modeling, Adaption and Personalization*, Springer, 2011, pp. 231–242. doi:10.1007/978-3-642-22362-4_20.

[24] H.-M. Park and C.-W. Chung, An efficient MapReduce algorithm for counting triangles in a very large graph, in: *Proceedings of the 22nd ACM CIKM13*, ACM, 2013, pp. 539–548.

[25] X. Sun and H. Lin, Topical community detection from mining user tagging behavior and interest, *JASIST* **64**(2) (2013), 321–333. doi:10.1002/asi.22740.

[26] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang and Z. Su, Arnetminer: Extraction and mining of academic social networks, in: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2008, pp. 990–998. doi:10.1145/1401890.1402008.

[27] X. Wang and A. McCallum, Topics over time: A non-Markov continuous-time model of topical trends, in: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2006, pp. 424–433. doi:10.1145/1150402.1150450.

[28] X. Wei and W.B. Croft , LDA-based document models for ad-hoc retrieval, in: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2006, pp. 178–185.

[29] J. Xie, S. Kelley and B.K. Szymanski, Overlapping community detection in networks: The state-of-the-art and comparative study, *ACM Comput. Surv.* **45**(4) (2013), 43. doi:10.1145/2501654.2501657.

[30] Z. Xu, Y. Ke, Y. Wang, H. Cheng and J. Cheng, A model-based approach to attributed graph clustering, in: *SIGMOD Conference*, 2012, pp. 505–516.

[31] J. Yang, J. McAuley and J. Leskovec, Community detection in networks with node attributes, in: *2013 IEEE 13th International Conference on Data Mining (ICDM)*, IEEE, 2013, pp. 1151–1156. doi:10.1109/ICDM.2013.167.

[32] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun and Z. Chen, CQArank: Jointly model topics and expertise in community question answering, in: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, ACM, 2013, pp. 99–108. doi:10.1145/2505515.2505720.

[33] H. Yin, B. Cui, H. Lu, Y. Huang and J. Yao, A unified model for stable and temporal topic detection from social media data, in: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, IEEE, 2013, pp. 661–672.

[34] H. Zhang, B. Qiu, C.L. Giles, H.C. Foley and J. Yen, An LDA-based community structure discovery approach for large-scale social networks, in: *ISI*, 2007, pp. 200–207.

[35] J. Zhang, M.S. Ackerman and L. Adamic, Expertise networks in online communities: Structure and algorithms, in: *Proceedings of the 16th International Conference on World Wide Web*, ACM, 2007, pp. 221–230.

[36] T.C. Zhou, M.R. Lyu and I. King, A classification-based approach to question routing in community question answering, in: *Proceedings of the 21st International Conference Companion on World Wide Web, WWW '12 Companion*, ACM, New York, NY, USA, 2012, pp. 783–790. doi:10.1145/2187980.2188201.