UNIVERSITY OF NICE - SOPHIA ANTIPOLIS

DOCTORAL SCHOOL STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

P H D T H E S I S

to obtain the title of

PhD of Science

of the University of Nice - Sophia Antipolis

Specialty : COMPUTER SCIENCE

Defended by

Zide MENG

Temporal and semantic analysis of user-generate-contents sites

Thesis Advisor: Fabien GANDON and Catherine FARON-ZUCKER

prepared at INRIA Sophia Antipolis, WIMMICS Team

defended on February 14, 2013

Jury :

Reviewers :	A B	-	CNRS (CREATIS)
	C D	-	McGill University
Advisor :	E F	-	INRIA (Asclepios)
President :	G H	-	INRIA (Asclepios)
Examinators :	I J	-	Centre Antoine Lacassagne (Nice)
	K L	-	University of North Carolina
	M N	-	Université Catholique de Louvain
Invited :	O P	-	DOSISoft S.A.

Acknowledgments

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.

Contents

CHAPTER 1

Introduction

Contents

1.1 Context

One of the significant changes of the Web was a move from Web 1.0 to Web 2.0. A main attribute of Web 2.0 is that it allow users to interact and collaborate with each other in a social media platform as creators of user-generated content(Moens 2014) in a virtual community. In contrast, people are only limited to the passive viewing of content in Web 1.0. Examples of Web 2.0 sites including social networking sites, blogs, forums, video sharing sites, etc. Web 2.0 is actually consist of a combination of user and the web, but it is not limited to relationship between users, it is rather built on the common interest among users. Therefore, it is crucial to study both user and user-generated content and discover useful information from them. It involves social network analysis(SNA). e.g. overlapping community detection. It also related to social media mining. e.g. Topic detection from

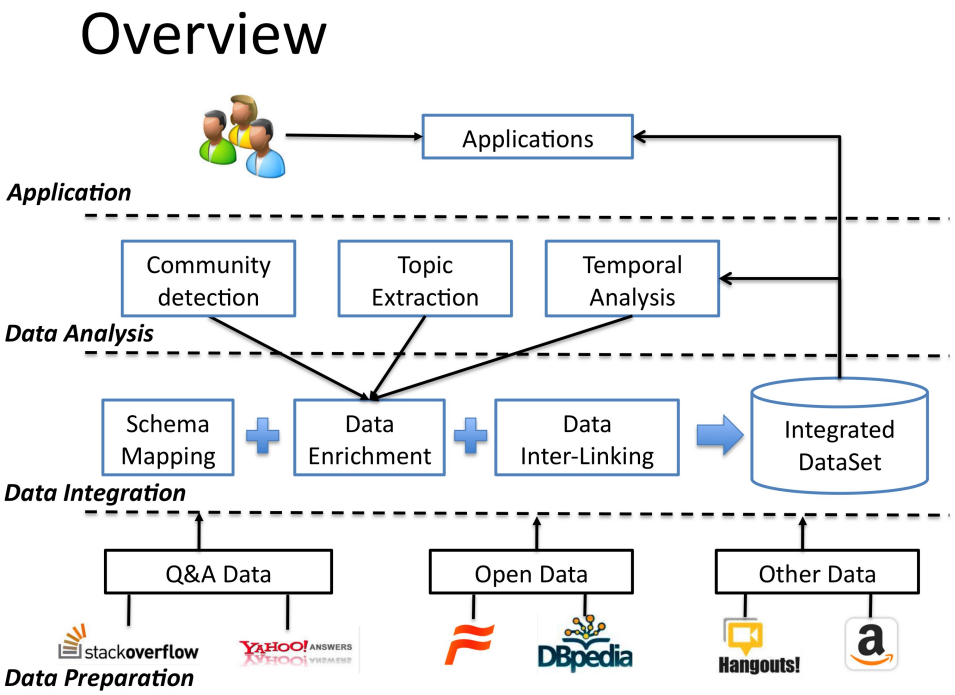


Figure 1.1: The overview of framework discussed in this thesis

user-generated content. Besides, users' behavior changing over time. It is also important to discovery those temporal information.

In another side, the web was evolving from a Web of Document to a Web of Data. According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries".[1]. However, User-generated contents on the Web are normally unstructured and isolated. Therefore, it is essential to formalize these information and transform them into knowledge.

In this thesis, we proposed a framework, which combine social network analysis, social media mining and semantic web technologies, to manage user-generated content websites. Figure 1.1 show the overview of proposed framework discussed in this thesis.

[1] https://www.w3.org/2001/sw/ (accessed Feb 2016)

1.2 Our Scenario

We mainly verify our framework on question-and-answer sites(Q&A sites), which is a typical User-generated Content website. Q&A sites initially aimed at enabling users to ask questions to a community of experts. Since these user-generated contents, which are questions and answers in this case, 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 (Anderson 2012). 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.

Regarding to the framework, firstly, we use designed schema to formalize all of the meta information of Q&A site. secondly, we mainly conduct three kind of analysis on this original dataset. we then integrate the results with the original dataset. thirdly, based on this integrate dataset, we provide several applications, such as, question recommendation, expert detection and user life-cycle management. This is the basic logic of the proposed framework.

The reason why we conduct the three kind of analysis is because that 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. What's more, Both user and topic are changing over time, detecting such temporal dynamics are important.

Detecting this information can contribute to the question routing problem (Li 2010a)(Zhou 2012), 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

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

1.3 Research Question

We summary all the research questions which the thesis will address and answer in this section.

RQ1. How to formalize user-generated content?

The information in User-generated content are unstructured. A key issue is how to formalize them. In addition, how to formalize the detected latent information have to be concerned.

RQ2. How can we identify the common topics binding users together?

On user-generated content websites, user are normally creating information on their interested topics. It is important to find topics from the content which user generated.

RQ3. How can we detect topics-based overlapping communities?

We address the problem of overlapping community detection. Different with traditional graph-structure based method, we try to solve this problem based on topic modeling. Therefore, topic can be directly used to interpret communities. Another reason is that, regarding to our scenario, 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. Therefore, 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.

RQ4. How can we extract topics-based expertise and temporal dynamics?

We address the problem of expert detection and temporal dynamic analysis together with topic modeling.

RQ5. How can we generate a semantic label for topics?

As our work are mainly based on topic, and a topic is consist of a bag of words. Therefore, it is essential to automatically generate a abstract label for each topic to describe the key meaning of each topic.

1.4 Contributions

The major contributions of this thesis are as follows.

- To address the research question **RQ1**, we present a prototype system to manage the two main resources in CQA sites: users and contents. We also present a vocabulary used to formalize the detect information
- To address the research question **RQ2**, we present a topic tree distribution method to extract topic from tags. We also proposed a first-tag enrichment method to enrich question which only have one or two tags. We show the effectiveness and efficiency of our topic extraction method.
- To address the research question **RQ3**, based on our topic extraction method, we present a method to assign user to different topics in order to detect overlapping communities.

- To address the research question **RQ4**, we present a joint model to extract topic based expertise and temporal dynamics from user-generated content. We also proposed a post processing method to model user activity information. Traditionally, these information have been modeled separately.

- To address the research question **RQ5**, we present a method using dbpedia to genrate an abstract label for a bag of words

1.5 Thesis Outline

This thesis contains a background and state of the art of related literature, an approach to detect topics from tags, an approach to detect overlapping communities, an approach to detect expertise and activities. The chapters in the rest of this thesis are organized as follows:

- Chapter 2 provides a background of all related topics, and the state of the art on community detection, topic modeling, expert detection and temporal analysis. We identify the research trends in the related areas, and outline the focus of this thesis.

- Chapter 3 describe QASM (Question & Answer Social Media), a system based on social network analysis (SNA) to manage the two main resources in CQA sites: users and contents. We also present the QASM vocabulary used to formalize both the level of interest and the expertise of users on topics.

- Chapter 4 describes an efficient approach for extracting data from QA sites in order to detect communities of interest. We also present a method to enrich a more general tag to questions which only have one or two tags. Then we compare three detection methods we applied on a dataset extracted from the popular QA site StackOverflow. Our method based on topic modeling and user membership assignment is shown to be much simpler and faster while preserving the quality of the detection.

- Chapter 5 describes a probabilistic graphical model to jointly model topics, expertises, activities and trends for question answering web application. we performed experiments with real-world data to confirm the effectiveness of our joint model, studying the usersâ behaviors and topics dynamics on a dataset extracted from the popular question-answer site StackOverflow.

- Chapter 6 describes an approach to automatically generate a label for topics, which are a bag of words. We conduct a user study to compared different algorithms to generate the label.

- Chapter 7 summarizes our contributions and describes our perspectives.

1.6 Publications

The publications resulted from this thesis are as follows:

- Journal

  1. Zide Meng, Fabien L. Gandon, Catherine Faron-Zucker, Ge Song: Detecting topics and overlapping communities in question and answer sites. Social Network Analysis and Mining 5(1): 27:1-27:17 (2015)

  2. Zide Meng, Fabien L. Gandon, Catherine Faron-Zucker: Overlapping Community Detection and Temporal Analysis on Q&A Sites. Web Intelligence and Agent Systems 2016. (submitted)

- Conference Paper

  1. Zide Meng, Fabien L. Gandon, Catherine Faron-Zucker: Joint model of topics, expertises, activities and trends for question answering Web applications. Web Science 2016. (submitted)

  2. Zide Meng, Fabien L. Gandon, Catherine Faron-Zucker: Simplified detection and labeling of overlapping communities of interest in question-and-answer sites. IEEE/WIC/ACM Web Intelligence 2015

3. Zide Meng, Fabien L. Gandon, Catherine Faron-Zucker, Ge Song: Empirical study on overlapping community detection in question and answer sites. IEEE/ACM ASONAM 2014: 344-348

4. Zide Meng, Fabien L. Gandon, Catherine Faron-Zucker: QASM: a QA Social Media System Based on Social Semantic. International Semantic Web Conference (Posters Demos) 2014: 333-336

CHAPTER 2

Background

Contents

2.1 Introduction

In this chapter, we review the related topics for the background knowledge for this thesis and provide a state of the art review of related literature. First, We introduce

the background of social web and semantic web. Second, we discuss the state of the art work related to our research question. Furthermore, we give a brief introduction about the funding project. Finally, we discuss the research trends and challenges, and the focus of this thesis.

2.2 Social Semantic Web

2.2.1 Social Web

The term "social Web" was coined by Howard Rheingold for this network in 1996. Rheingold was quoted in an article for Time on his website "Electric Minds", described as a "virtual community" that listed online communities for users interested in socializing through the Web, saying that "The idea is that we will lead the transformation of the Web into a social Web" (Rheingold 2000). According to the World Wide Web Consortium (W3C), the Social Web is a set of relationships that link together people over the Web. [1]. The social web are designed and developed to support social interaction (Porter 2010) on the web. These on-line social interaction including online shopping, blogs, forums, video sharing and social networking websites. Today, hundreds of millions of Internet users are using thousands of social websites to connect with friends, discover news and to share user-generated content, such as blogs, photos, microblogs, videos. By the end quarter in 2008, Facebook reported 67 million members, YouTube had more than 100 million videos and 2.9 million user channels (Watson 2008), and these numbers are consistently growing.

2.2.1.1 Web 2.0

One of the significant change for the World Wide Web is moving from Web 1.0 to Web 2.0. The term Web 2.0 is initially coined by Darcy DiNucci in 1999

[1] https://www.w3.org/2005/Incubator/socialweb/XGR-socialweb-20101206/(accessed Feb 2016)

Web 1.0		Web 2.0
DoubleClick	-->	Google AdSense
Ofoto	-->	Flickr
Akamai	-->	BitTorrent
mp3.com	-->	Napster
Britannica Online	-->	Wikipedia
personal websites	-->	blogging
evite	-->	upcoming.org and EVDB
domain name speculation	-->	search engine optimization
page views	-->	cost per click
screen scraping	-->	web services
publishing	-->	participation
content management systems	-->	wikis
directories (taxonomy)	-->	tagging ("folksonomy")
stickiness	-->	syndication

Figure 2.1: A comparison of examples of Web 1.0 and Web 2.0, as in (O'really 2009)

(DiNucci 2012) and become popular by Tim O'Reilly in 2005 (O'really 2009). User can interact and collaborate with each other and create user-generated content in online community on Web 2.0 site, while user are only allowed to browse content on Web 1.0 site. A comparison of examples of traditional Web 1.0 and Web 2.0 shown in Figure 2.1. Popular examples of Web 2.0 sites are Facebook (social networking service), Twitter (a microblog), Youtube (A video-sharing website), Reddit (A user-generated news website).

With the evolution of web development technology, such as Asynchronous JavaScript and XML (AJAX), Rich Internet Applications(RIA), Cascading Style Sheets (CSS), etc. Web 2.0 allows users to create and share richly-typed user-generated content with each other more easily. (Passant 2009) argue that there are two main principles in Web 2.0, the first one is the *"Web as a platform"*, which implies the migration from traditional desktop applications (email clients, office suites, etc.) to Web-based applications. The second one is the *"architecture of participation"*, which represents how user change from data consumer to data producer in Web-based applications. For more design principles of Web 2.0 websites, we suggest the reader to (O'really 2009)

2.2.1.2 User-generated content

The OECD(Web 2007) define that User-generated content (UGC) have following requirements: 1) a content which is made publicly available through internet. 2) boasting a certain level of creativity and maybe the most important point. 3) contents created outside of professional practices. User-generated content (UGC) could be any form of content such as blogs, photos, Q&A, forums, tweets, posts, videos which created by users of an online social media websites. (Moens 2014). The reasons why people contribute to user-generated content are mainly: connecting with people, self-expression and receiving recognition. For example, Users connect with friends on Facebook, Users express themselves on Twitter[2]. Users share their photos on Flickr[3]. Users ask and answer computer programming related questions on StackOverflow[4]. Nevertheless, there are some issues (Balasubramaniam 2009) about UGC, such as, trust problem, since the content are written outside of non-professional; privacy problem, since the content often contain many very private information; copyright problem, more attention should be put on protecting the right of user-generated content. For more detailed information about the driver and evolution of UGC, the commercial influence of UGC, we recommend the readers to (Balasubramaniam 2009) and (Smith 2012).

2.2.1.3 Q&A sites

Q&A sites, also refer to community question answering (CQA), initially aimed at enabling users to ask questions to a community of experts. Since these user-generated contents, which are questions and answers in this case, 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. For example[5], the first

[2] https://twitter.com/(accessed Feb 2016)
[3] https://www.flickr.com/ (accessed Feb 2016)
[4] http://stackoverflow.com/ (accessed Feb 2016)
[5] https://en.wikipedia.org/wiki/Comparison_of_Q&A_sites (accessed Feb 2016)

potential Q&A site Naver Knowledge iN [6] launched in 2002 in Korean, has accumulated 70 million quesitons and answers, and continued to receive over 40,000 questions and 110,000 answers per day(Sang-Hun 2007). Baidu Knows[7] and Zhihu [8] are the most popular Q&A sites in China. It is reported [9] that the number of registered users of Zhihu had exceeded 10 millions by the end of 2013, and reached 17 millions in May 2015 with 250 million monthly page views. Yahoo Answers, lunched in 2005, offers Q&A sites localized to 26 countries and it is said(Harper 2008) that Yahoo Answers is estimated to have 18 millions unique visitors monthly in September 2007.

We compare the traditional keyword-based search engine and question answer site. In search engine, people should choose some keywords to describe their problem, then look for related information to solve the question. In question answer site, people should post their question and wait for experts to solve it. Table 2.1 compare the difference between Q&A and Search Engine.

	Problem define	Timely	Precise	Solution
Q&A	Well organized questions and background info.	Until someone answer it	Specified on the question	directly get the answers
Search Engine	Well chosen keywords or short question	immediately get relevant info.	Not specified on the question	Need analyze the info

Table 2.1: Comparison of the Q&A and Search Engine

In a Q&A site, people need to provide very detail information about their questions, in order to let other user understand it. While in a search engine, people have to wisely choose search keywords to look for solutions, the quality of keywords sometimes largely influence the results. When we pose a question to a Q&A site, it takes time to attract other user and get the answers, but the search engine can immediately return most relevant information. Once people get answer from Q&A site, normally it is very specified to the question and very precise. So Q&A site can solve very complicated questions. In search engine, people can get very relevant information

[6] http://naver.com (accessed Feb 2016)
[7] http://zhidao.baidu.com/ (accessed Feb 2016)
[8] http://www.zhihu.com/ (accessed Feb 2016)
[9] https://en.wikipedia.org/wiki/Zhihu (accessed Feb 2016)

about the keywords we provide. However, in sometime, these information are very general and not specified to the question, and people have to find the solutions from these information by themselves.

On one hand, Q&A sites have become huge repositories of question-answer content which support highly valuable and highly reusable knowledge (Anderson 2012), (Shah 2010). 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. Thus, We could believe these user-generated content are normally with high quality as they are generated by people with very strong domain knowledge and expertise. We list key features of some famous Q&A sites in table 2.2.

	Category	Reward to answerer	Question Tag	Vote	Best Answer
Yahoo Answer	Many Categories	Level and Points	no	only on answers	yes
StackOverflow	Computer Science	Reputation system	yes	Both on questions and answers	yes
Baidu knows	Many Categories	Level and Coins	no	Both on questions and answers	yes

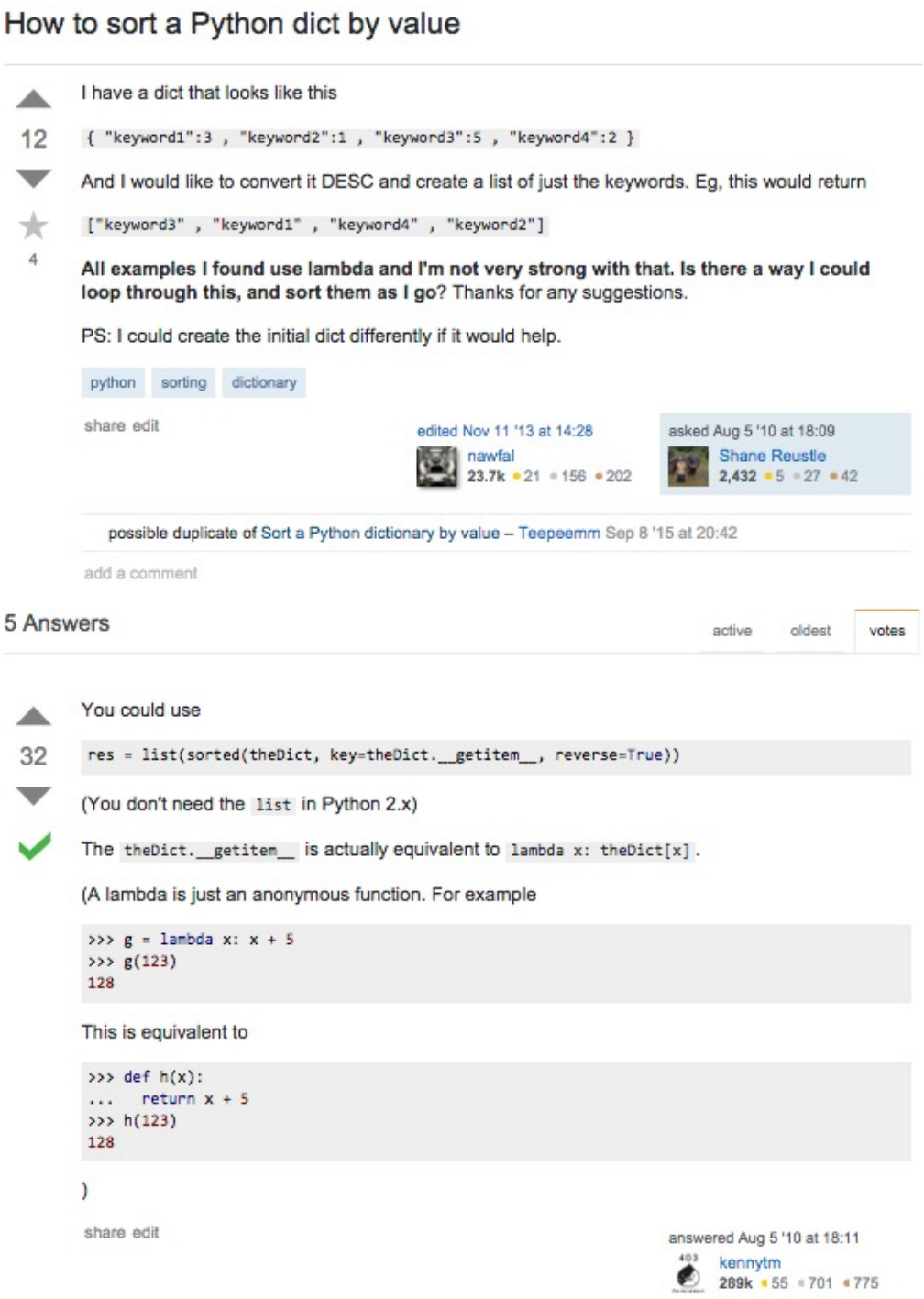
Table 2.2: Key features of famous Q&A sites

StackOverflow is the most popular Q&A site that focus on computer programming topics. Figure 2.2 shows a example of question and answer on StackOverflow.

2.2.2 Semantic Web

According to W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries"[11]. Tim Berners-Lee (Berners-Lee 2001) use this term to refer a web of data that can be processed by machines. It is a change from web of document to a web of data. People generate and consume huge amount of data all most every day. However, these data are kept by each application or each website, and people have to manager and process the interaction of these information by themselves. For example, in order to make a trip plan, a user should check different website including

[11] https://www.w3.org/2001/sw/ (accessed Feb 2016)



How to sort a Python dict by value

12

I have a dict that looks like this

```
{ "keyword1":3 , "keyword2":1 , "keyword3":5 , "keyword4":2 }
```

And I would like to convert it DESC and create a list of just the keywords. Eg, this would return

```
["keyword3" , "keyword1" , "keyword4" , "keyword2"]
```

4

All examples I found use lambda and I'm not very strong with that. Is there a way I could loop through this, and sort them as I go? Thanks for any suggestions.

PS: I could create the initial dict differently if it would help.

python sorting dictionary

share edit

edited Nov 11 '13 at 14:28
nawfal
23.7k • 21 • 156 • 202

asked Aug 5 '10 at 18:09
Shane Reustle
2,432 • 5 • 27 • 42

possible duplicate of Sort a Python dictionary by value – Teepeemm Sep 8 '15 at 20:42

add a comment

5 Answers

active oldest votes

32

You could use

```
res = list(sorted(theDict, key=theDict.__getitem__, reverse=True))
```

(You don't need the `list` in Python 2.x)

The `theDict.__getitem__` is actually equivalent to `lambda x: theDict[x]`.

(A lambda is just an anonymous function. For example

```
>>> g = lambda x: x + 5
>>> g(123)
128
```

This is equivalent to

```
>>> def h(x):
...   return x + 5
>>> h(123)
128
```

)

share edit

answered Aug 5 '10 at 18:11
kennytm
289k • 55 • 701 • 775

Figure 2.2: A example of question and answer on StackOverflow [10]

flight, hotel, weather, train schedule and so on. It is even not easy for human to integrate them. For example, a small change of flight may cause user to check and change all the other reservations. It is also not possible for application to manage all these information from different website. However, instead of web of document, if we have a web of data, then it is possible for application to process and integrate them together. So, the main attribute of Semantic Web is to enable content provides not only publish human-readable web document, but also machine-readable data. Thus, with this vision, the Semantic Web will allow applications to process data from different source the same way as people gathering information from different web pages. Later in 2006, Tim Berners-Lee(Berners-Lee 2006) proposed the Linked Data principles for publishing structured data on the Semantic Web. It is a method to share Semantic Web data using the Web architecture (Bizer 2011). An important development in this context is the W3C Linking Open Data (LOD) [12]. Figure 2.2.2 shows the LOD cloud diagram[13]. It shows the data sets that has been published as Linked Data. As of August 2011, the LOD cloud contains 295 data sets classified into 7 domains totaling 31,634,231,770 triples [14]. There are 20 data sets and 134,127,413 triples in the domain of User-generated Content, which only takes 0.42%.

In the following subsections, we briefly introduce the RDF data model which used to represent data on the Semantic Web and the related vocabulary to formalize social media dataset. For more details about the objectives and goals of the Semantic Web, we recommend the readers to (Feigenbaum 2007) and (Berners-Lee 2001).

[12] https://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData (accessed Feb 2016)

[13] Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/

[14] http://lod-cloud.net/state/ (accessed Feb 2016)

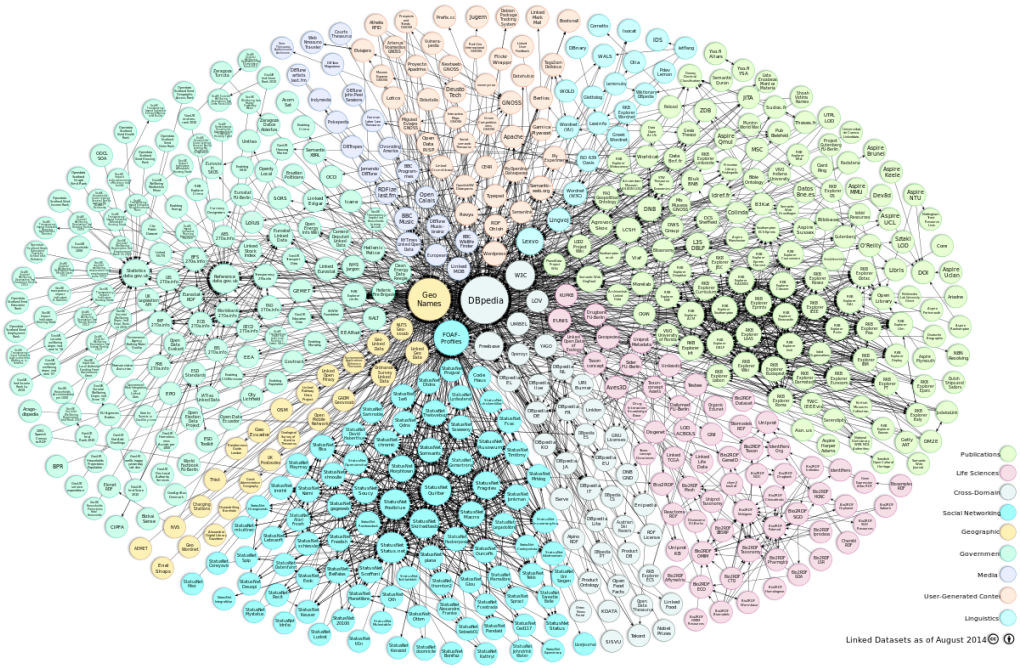


Figure 2.3: Linked Open Data cloud diagram.

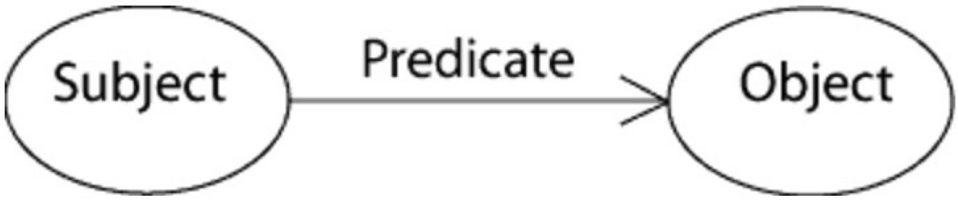


Figure 2.4: The Graph Data Model of RDF.

2.2.2.1 RDF

The Resource Description Framework (RDF) data model is used to describe resources with triples (subject, predicate, object) that can be viewd as "a natural way to describe the vast majorrity of the data processed by machines". It commonly refer to " the subject, the predicate and the object triple". The graph data model[15] is show in Figure 2.2.2.1.

The subject represents the described resource. The predicate represents the property

[15]Resource Description Framework (RDF):Concepts and Abstract Syntax https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-Concepts (accessed Feb 2016)

used to describe the resource. The object represents the value of the property for the described resource. Then people can easily define and describe any resources with this model. For example, to formalize an question [16] from a popular question answer site Stackoverflow, we produce the following triple:

- (<http://stackoverflow.com/users/1214235/kingrauk>, <http://rdfs.org/sioc/ns#owner_of>, <http://stackoverflow.com/questions/16772071/sort-dict-by-value-python>)

. With respectively:

- <http://stackoverflow.com/users/1214235/kingrauk> is the URI that identifies the user who create the question
- <http://rdfs.org/sioc/ns#owner_of> is the URI that identifies the property *owner_of*
- <http://stackoverflow.com/questions/16772071/sort-dict-by-value-python> is the URI that identifies the user

Similarly, we can create another triples to formalize the answers. ()

- (<http://stackoverflow.com/questions/16772071/sort-dict-by-value-python/16772088#16772088>, <http://rdfs.org/sioc/ns#reply_of>, <http://stackoverflow.com/questions/16772071/sort-dict-by-value-python>)

. With respectively:

- <http://stackoverflow.com/questions/16772071/sort-dict-by-value-python/16772088#16772088> is the URI that identifies the answer to the question

[16] http://stackoverflow.com/questions/16772071/sort-dict-by-value-python (accessed Feb 2016)

- <http://rdfs.org/sioc/ns#reply_of> is the URI that identifies the property *reply_of*
- <http://stackoverflow.com/questions/16772071/sort-dict-by-value-python> is the URI that identifies the question

For instance, Figure **??** represent the graph of two previous description about the question.

2.2.2.2 Ontology

Ontology is "a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application"(Liu 2009).

RDF enable people to define properties and descriptions to formalize domain knowledge. However, RDF does not provide the basis primitives to define properties, classes. Thus, it is necessary to define an ontology for specific domain knowledge.

sioc

time

distribution

2.3 ANR project

Over the past 15 years, along with the success of the Social Web, online communities have progressively produced massive amounts of user-generated content collaboratively.

While some of these communities are highly structured and produce high-quality content (e.g., open-source software, Wikipedia), the level of discussions found

within less structured forums remains highly variable. Coupled with their explosive growth, the lower quality of online forums makes it hard to retrieve relevant and valuable answers to user search queries, and subsequently diminishes the social and economic value of this content.

The objective of OCKTOPUS is to increase the potential social and economic benefit of this user-generated content, by transforming it into useful knowledge which can be shared and reused broadly.

One of the most visible and easily-understandable output of the project is a demonstration platform which can be used to input a newly-formulated question, search online forums for a similar already-answered question, and display a unique user-generated answer associated with these similar questions.

This demonstration platform is built around the idea that finding relevant high-quality answers can be broken down in two steps:

Triage user-generated content to extract gold (knowledge structured as pairs of questions and answers) from ore (random discussions) Given a newly-formulated question, retrieve relevant similar questions within the gold. OCKTOPUS therefore investigates mainly, to what extent can newer data mining techniques based on the proper assessment 1) of the organizational traits of online communities, 2) of the tree-structure of online discussions, and 3) of the temporal dynamics of large typed semantic user-user graphs, help improve the automatic classification and triage of unstructured online content?

2.4 Overlapping 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.

2.4.1 Graph-based Methods

A first and direct solution is to extract an implicit network structure (such asa 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 (Xie 2013)(Ahn 2010). 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 indicate that they have answered the same questions many times. They cannot provide the information 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 (Xie 2013) address this problem.

2.4.2 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, we first have to define the distance between user's interests and the definition of this distance 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 (Xu 2012)(Gargi 2011), group users according to their features. They do not take the network structure into consideration. Moreover, some clustering algorithms normally output hard-partition

communities, one user can only be assigned to one 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. (Chang 2013) 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, (Chang 2013) does not give any details on how to compute the topic tag distribution and user topic distribution, while we do.

2.4.3 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, (Zhang 2007a) transform links to binary node attributes, then use a LDA-based model to detect communities. (Sun 2013) 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. (Tang 2008) 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 (Yang 2013a) 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, (McDaid 2010) and (Lancichinetti 2011) also use statistic 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.

2.4.4 Short Summary

Table 2.3 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 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 (Yang 2013a). In addition, our proposed method is much simpler and faster than LDA-based methods while preserving the quality of the detection.

Table 2.3: Comparison of the main approaches and our method

	uses nodes	uses links	overlap	membership	topic
Graph-based	no	yes	few	few	no
Clustering meth-ods	yes	no	few	few	no
LDA-based	yes	yes	yes	yes	yes
Our-method	yes	yes	yes	yes	yes

2.5 Expert Detection

Research 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. (Zhang 2007b) 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 statistic measures, for instance the number of best answers user provided, the Z-score measure uses both the number of questions and the number of answers a user posted. Similarly, (Jurczyk 2007) use the HITS algorithm to discover authorities users. (Bouguessa 2008) propose a model based on Indegree, which is the number of best answers provided by users, to discovery experts. (Li 2010a) propose a probability model to estimate users' expertise for question routing task.

Rather than detecting global experts, another kind of works uses topic models to detect topic level experts. (Guo 2008b) proposed a generative model by leveraging the category information of questions on certain Q&A sites. (Yang 2013b) jointly model topics and expertise by integrating a Gaussian Mixture Model to capture vote information. (Chang 2013) propose a spectral clustering based topic model. (Ma 2015) 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. (Ji 2013) combine topic models outputs and statistic features and apply a pairwised learning to obtain a ranked model and recommend expert users for a question. (Pal 2011) apply machine learning algorithms to identify experts from their early behavior. (Anderson 2012) perform an in-depth study of StackOverflow[17] and show that expert users tend to answer questions more quickly and gain high reputation by higher activity. Their work is based on features extraction and machine learning al-

[17] http://www.stackoverflow.com/ (accessed Feb 2016)

gorithms 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.

2.6 Temporal Analysis

there is an increasing research interest for the temporal modeling of online communities and several methods have been proposed. (Wang 2006) jointly model topics and time label by assuming that words and time stamps are both generated by latent topics. (Yin 2013) proposed a PLSA-based(Hofmann 1999) model to separate temporary topics from stable topics. (Hu 2014) jointly model latent user groups and temporal topics to detect group-level temporal topics.

2.7 Discuss and Summary

Compared with these latest works, our model not only captures topics and expertise, it also can detect topic dynamics both at the global community level and at the individual user level. Besides, we propose a post-process method to solve a common problem in LDA based models.

There are two aspects related to Q&A sites.

- Social aspect: A large number of people are very active and keep contributing answers to these sites. Most of them are more likely to answer questions about topics in which they are interested and specialized. So we want to find the interest groups of users in Q&A sites. Community detection is a fundamental research topic for social network analysis. Many community detection algorithms have been developed to find sub-structures in social networks. Q&A sites are also social networks. However, unlike friendship networks such as

Facebook, there are no explicit relationships between people on Q&A sites. Besides people are not aware of who they are interacting with, and normally they do not maintain a solid relationship. People are more like isolated nodes grouped by interests. So interest groups are an important implicit sub-structure in such social sites. Moreover, people have multiple interests and therefore belong to several interest groups. Therefore an important question of this aspect is the ability to detect overlapping communities or interest groups.

- Content aspect: Another important resource in Q&A sites are question answer pairs. These questions cover different topics, and user who ask or answer a question can reflect that he/she is interest in a topic. Therefore, The topics of questions and interests of groups are consistent. We want not only to detect interest groups, but also to find the topic of each interest group. Topic extraction is a critical research problem in text analysis. Many topic extraction methods have been proposed to cluster texts by its topics. The reason why we need to do content analysis is that it could enable applications such as similar questions recommendation and question routing, which are very important features in Q&A scenario.

We list and analysis some work which have been done in Q&A sites.

2.7.1 Expert detection

(**?**) 's work address a core problem in Q&A site. They argue that most of related work in expert finding are based on link analysis while ignoring the topical similarity among users and user expertise and user reputation. They proposed a topic-sensitive probabilistic model to find experts in Q&A sites. This model is based on LDA which is widely used in topic modeling. Then it generate a topic similar graph based on the result of topic model. Then a PageRank algorithm is applied to find the experts. They compared their work with many state of the art link analysis algorithm and show gain in the experiment.

(**?**) is not specified on Q&A community. It focus on a more boarder website: help-seeking websites. it tests pagerank and hits algorithm to detect expert in such websites. Pagerank and Hit are well known authority algorithm in directed graph analysis. By constructing users into direct graph, then they could apply this algorithm to find the most import node in the graph.

(**?**) proposed a temporal pattern based expert detecting method. The temporal pattern is based on the reputation system of Q&A site where user has high reputation is considered as an expert. It use an supervised learning algorithm to distinguish expert from normal user. The limitation of this work is it can not find out in which topics are people specialized.

(**?**) proposed a method using link analysis techniques to find a list of expert user based on the in-degree of authority, which is computed by the number of best answers provide by a user. Then they use Bayesian Information Criterion(BIC) to estimated the authority score of a user. Therefore, experts are chosen according to its authority score. Expertiment is taken on Yahoo Answer.

2.7.2 Question Routing

(Guo 2008a) try to solve question routing problem, which we category it as Q5. They proposed a LDA-like probability model to find the latent topic of users and latent topic of questions and answers. Then based on these topic information, they could route a new question to a user which have the same topic distribution.

(**?**) proposed a Topic Expertise Model which is a LDA-like probability model combined with a GMM model to detect experts in Q&A sites. The probability model is mainly used for extracting topics from tags and words in Q&A, it actually contains two LDA process: 'user-topic-tag' and 'user-topic-content' . The GMM(Gauss Mixture Model)is used for analyzing users expertise on each topics. The output include topic-tag distribution, user-topic distribution, topic-word distribution, users' topic-expertise matrix. Then according to these output, it can give top tags of a topic, top

users of a topic, top experts of a topic. The experiments show it can outperform the state-of-the-art probability model in Q&A sites.

(**?**) proposed a recommendation model, which integrate topical expertise and availability of users, to recommend answerers and commenter to a question. It constructs a similarity matrix between tags, and runs spectral clustering algorithm over it. Then a cluster of tags can be viewed as a topic. But unlike LDA, spectral clustering can not output the topic-tag distribution which will limit the flexibility of afterwards application. The paper proposed a question-topic distribution, but it dose not mention how to compute it. So the conclusion that spectral clustering can out perform LDA is not clear. And spectral clustering is hard partition of tags, while LDA can give proportion that a tag belong to a topic.

2.7.3 Similar Question

(**?**) 's work investigate the general characteristic of stackoverflow dataset. A contribution of this work is to predict long-term value of a question. They find strong evidence that only 37% of favorites on a questions arrive within the time frame when the question is being answered. Actually the content in Q&A sites mainly server two kind of people. One is people who ask his question and the other is people who search question. So, if a question have a long-term value, it could be more possible to be searched again. We category this work into Q4 since finding out these questions could improve the result of searching similar question. They developed four categories of features for learning. It includes, 4 questioner features which are related to questioner's behaviors; 8 activity and Q/A quality measures which are extracted from questions and answers; 8 community process features which are related reputation of answers; and 7 temporal process features which are generated from the time information of Q/A behavior. Then they treat this problem as a binary classification task and use a machine learning technique to predict whether a question have a long last value. They compare their work with a baseline which only use upvote

	Expert	Routing	SimilarQ	Methods	Dataset	Topic
(**?**) 2005	no	no	yes	Probability Model	Naver[18]	no
(**?**) 2007	yes	no	no	PageRank,Hits	Forum	no
(Guo 2008a) 2008	no	yes	no	LDA based	StackOverflow	yes
(**?**) 2008,(**?**)2008	no	yes	yes	PLSA	Yahoo,Wenda	yes
(**?**) 2008	yes	no	no	Link Analysis	Yahoo	no
(**?**) 2010	yes	no	no	Supervise Learning	StackOverflow	no
(**?**) 2012	no	no	yes	Supervise Learning	StackOverflow	no
(**?**) 2012	yes	no	no	LDA based	StackOverflow	yes
(**?**) 2013	yes	yes	yes	LDA based	StackOverflow	yes
(**?**)2013	yes	yes	no	SpectralClustering	StackOverflow	yes

Table 2.4: Comparison of several works in Q&A sites. Expert denote 'Expert detection', Routing denote 'Question Routing', Similar denote 'Similar Question Finding', Methods denote 'Proposed algorithm', Dataset denote 'Experiment Data' Topic denote 'Topic Detection'

and downvote features.

(**?**) discuss methods for question retrieval that are based on using the similarity between answers. It proposed a translation-based retrieval model to find similar questions. The experiment shows that it's possible to find semantically similar questions with relatively litter overlapping words. They found question title can provide best performance for retrieving similar question. This work based on a tuition that most of the people don't check whether their question has been asked which will lead to a situation that there could be many semantically identical questions. Therefore, they use the similarity between answers to group similar questions. A translation model based algorithm is proposed to calculate the similarity between answers. For example, this model could give the similarity score between 'bmp' to 'jpg'. Experiment show that the model can out perform other language models and similarity metrics. Table 2.4 list the comparison of the above works.

(**?**) present probabilistic latent semantic analysis (PLSA) approach to compute the probability of a user answer a question. They actually build a user-interest-question model. PLSA and LDA are quite similar and both are topic models, and LDA could be viewed as an extension of PLSA. The experiment shows that topic features based similarity can outperform cosine distance based similarity. (**?**) also used PLSA to recommend questions.

CHAPTER 3

QASM: Questoin and Answer Social Media

Contents

3.1 Introduction

Community Question Answering (CQA) services provide a platform where users can ask expert for help. Since questions and answers can be viewed and searched afterwards, people with similar questions can also directly find solutions by browsing this content. Therefore, effectively managing these content is a key issue. Previous research works on this topic mainly focus on expert detection (Yang 2013b), similar question retrieval (Anderson 2012). In this paper, we describe QASM (Question & Answer Social Media), a system based on social network analysis (SNA) to manage the two main resources in CQA sites: users and contents. We first present the QASM

vocabulary used to formalize both the level of interest and the expertise of users on topics. Then we present our method to extract this knowledge from CQA sites. Our knowledge model and knowledge extraction method is an extension of our work presented in (**?**) on social media mining for detecting topics from question tags in CQA sites. Finally we show how this knowledge is used both to find relevant experts for routing questions (users interested and experts in the question topics) and to find answers to questions by browsing CQA content and by identifying relevant answers to similar questions previously posted. We tested QASM on a dataset extracted from the popular CQA site StackOverflow.

3.2 QASM System Description

3.2.1 Overview

Figure 3.1 presents an overview of QASM. We first use the SIOC ontology[1] to construct an RDF dataset from social media data extracted from a CQA site. Then we use social media mining techniques to extract topics, interests and expertise levels from this dataset. We formalize them with the QASM schema and enrich our RDF dataset with this knowledge. As a result, we provide an integrated and enriched Q&A triple store which contains both user interests, levels of expertise and topics learned from question tags. Finally, we linked our dataset with DBpedia (through named entity identification).

Based on the QASM RDF dataset, we can provide the users of the Q&A site with two services to find relevant experts for a question and to search for similar questions. We detail them in the following subsections.

Figure 3.1: Overview of QASM

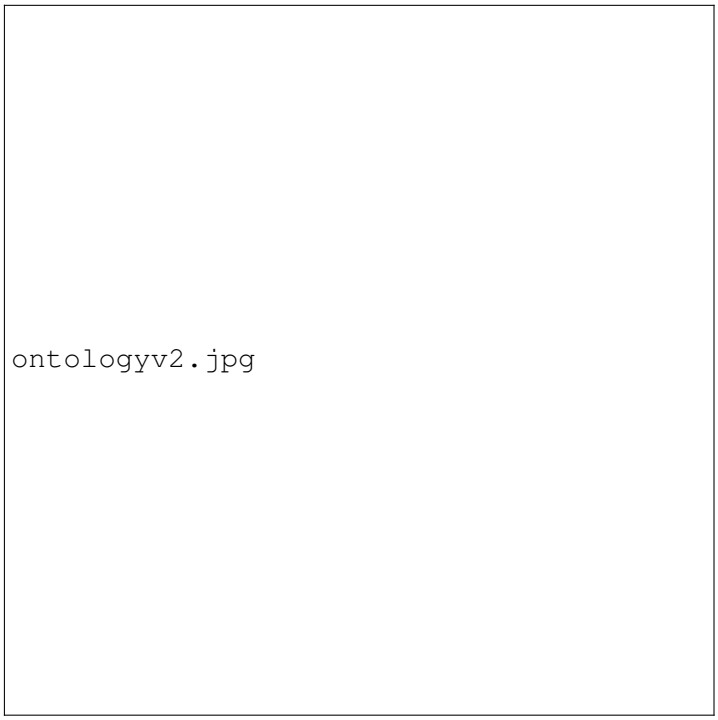


Figure 3.2: Overview of the QASM vocabulary

3.2.2 QASM Vocabulary

The QASM vocabulary[2] enables to model the level of user interests and expertise and topics of questions and answers from Q&A sites. Figure 3.2 provides an overview of it. It reuses both the SIOC ontology and the Weighting ontology[3].

- `qasm:Topic` represents a set of tags related to a specified topic. In our models, tags belong to instances of `qasm:Topic`, we also consider different tags have different weights for each topic.
- `qasm:WeightedObject` is used to describe the weight that a specified subject has with regard to a specified object. This class has four subclasses which represent question topics, users' interests, users' expertise and tag topics respectively. In fact, this class is used to model the distributions we extracted from the original data. For example, topic-tag distribution, user-interest distribution.
- `qasm:interestIn` is used to describe the user-interest distribution. This property is different from `foaf:interest` for its range. In FOAF people are interested in documents, while in QASM a user is interested in a topic to a certain degree (a weight).
- `qasm:expertiseIn` is used to describe the user-expertise distribution. A user has different weights for different topics.

3.2.3 Knowledge Extraction by Social Media Mining

Topics, interests and levels of expertise are implicit information in the available raw CQA data. We use social media mining techniques to extract this knowledge.

- Topics & User Interests In (**?**), we proposed a light-weight model to extract topics from question tags. The output of this model is a topic-tag distribution

[1] http://sioc-project.org/ontology
[2] It is available online at http://ns.inria.fr/qasm/qasm.html
[3] http://smiy.sourceforge.net/wo/spec/weightingontology.html

Table 3.1: Preliminary results on question routing

	100	500	1000	average	(**?**)
precision@10	0.021	0.0188	0.0187	**0.0195**	0.0167
precision@20	0.016	0.0134	0.0134	**0.0143**	0.0118

where each tag belonging to a topic is given a weight (probability) indicating to what extent the tag is related to the topic. A user answering a question acquires the tags attached to this question and can therefore be represented by a list of tags. Then we use the topic-tag distribution to compute a user-topic distribution indicating to what extent each user is related to a topic.

– User Expertise The users interested in a question may provide answers to it or comments to other answers. Each question or answer may get votes from other users and an answer may be chosen as the best answer. By exploiting the tags attached to a question and the topic-tag distribution, the users providing questions or answers with a high number of votes or the best answers can be considered as experts in the topics to which their questions belongs. Equation 3.1 defines how we use the vote information to compute users' levels of expertise. $E_{u,k}$ denotes the expertise of user u on topic k, m denotes the number of answers provided by user u, $P_{t,k}$ denotes the weight of tag t for topic k, Q_i and $A_{i,j}$ denote the votes on question i and its j^{th} answer, where A_j is the j^{th} answer provided by user u to question Q_i.

$$E_{u,k} = \sum_{i=1}^{m} P_{t,k} * log(Q_i) * log(A_{i,j}) \tag{3.1}$$

3.2.4 Experimental Evaluation

We first built an RDF dataset from Stackoverflow raw data which comprises 15327727 triples[4]. Then we randomly chose several questions and for each question we recorded 10 or 20 users provided by our system. Then for each question, we computed the proportion of the recorded users who actually answered it. Compared to (**?**), our results are much better.

[4]It is available online at https://wimmics.inria.fr/data

3.3 Conclusion and Future Work

We presented QASM, a Q&A system combining social media mining and semantic web models and technologies to manage Q&A users and content in CQA sites. There are many potential future directions for this work. We are currently considering constructing a benchmark for Q&A system based on our Stackoverflow dataset. In a near future we will also enrich the linking of QASM with the LOD which may help to improve question routing and similar question search.

CHAPTER 4

Overlapping Community Detection based on the LDA Model

Contents

4.1 Principle of our Approach

In Natural Language Processing (NLP), Latent Dirichlet Allocation (LDA) (Blei 2003) is a classical document 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 (Li 2010b), 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 4.1:

$$P(t|u) = P(t|z) * P(z|u) \tag{4.1}$$

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 Figure 4.1. 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 4.1. 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.

The intuition behind this model is that users choose their topics and topics generate

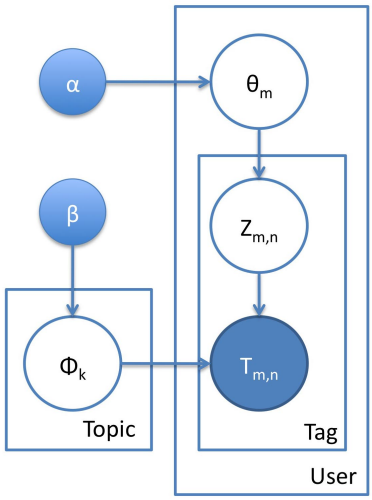


Figure 4.1: User-Topic-Tag (LDA) Model

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 $\phi(k) \sim$ Dir(β)

for user $m \in [1, M]$:

draw a user-topic distribution $\theta(m) \sim$ Dir(α)

for each tag $n \in$ user m's tag list, where $n \in [1, N_m]$, $m \in [1..M]$

draw topic $z_{m,n} \sim$ Multi($\theta(m)$)

draw tag $t_{m,n} \sim$ Multi($\phi(z_{m,n})$)

Table 4.1: 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 n^{th} tag in m's tag list
$t_{m,n}$	the specified tag in n^{th} position of m's tag list

We use the collapsed Gibbs Sampling method (Griffiths 2004) 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 4.2.

$$\begin{aligned} p(z_i = z_{m,n} | u = u_m, t = t_{m,n}, Z, U, T_{\neg i}) \\ \propto \frac{C^{z_{m,n}}_{u_m,\neg i} + \alpha}{\sum_{k=1}^{K} C^{k}_{u_m,\neg i} + K * \alpha} \\ \cdot \frac{C^{t_{m,n}}_{z_{m,n},\neg i} + \beta}{\sum_{t=1}^{V} C^{t}_{z_{m,n},\neg i} + V * \beta} \end{aligned} \tag{4.2}$$

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 equation 4.3 and 4.4:

$$\theta = \frac{C^{k}_{u} + \alpha}{\sum_{k=1}^{K} C^{k}_{u} + K * \alpha} \tag{4.3}$$

$$\phi = \frac{C^{t}_{k} + \beta}{\sum_{t=1}^{V} C^{t}_{k} + V * \beta} \tag{4.4}$$

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.

4.2 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.

A first result is the probability for each tag to belong to each topic. This is shown in Table 4.2. The second result is the probability for a user to belong to different topics

of interest. This is shown in Table 4.3.

Table 4.2 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 concerned with iphone-development, its top 3 tags are 'iphone', 'objective-c' and 'cocoa' which are very relevant to it.

Table 4.2: Top 10 related tags for detected topics of interest

topic1	**topic2**	**topic3**	**topic4**
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)

topic5	**topic6**	**topic7**	**topic8**
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 4.3 shows six randomly chosen users and their top 10 tags. The first row con-

tains 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.

Table 4.3: 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)

4.3 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 more simple method to detect topics and overlapping communities to solve the first problem: the TTD method is presented in Section 5.1. 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 **??**.

CHAPTER 5

Tag based topic extraction

Contents

5.1 Topic Trees Distributions (TTD)

5.1.1 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 $U = \{u_1, u_2...u_n\}$ be the set of users, $Q = \{q_1, q_2...q_m\}$ the set of questions and $T = \{t_1, t_2...t_v\}$ the set of tags. We aim at (1) extracting topics distribution $Topic = \{topic_1, topic_2...topic_k\}$ from T, and for each $topic_k$, defining $topic_k = \{p_{k1}, p_{k2}...p_{kv}\}$ 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 $I_i = \{I_{i1}, I_{i2}...I_{ik}\}$ where I_{ik} denotes the probability of u_i to be interested in $topic_k$.

5.1.2 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, 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 ex-

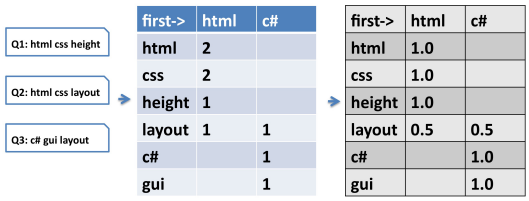


Figure 5.1: Example of computing a first-tag distribution

ample, as shown in Figure 5.1, 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 5.1, 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.

$$\begin{aligned} p(first\_tag|tag) &= \frac{p(first\_tag, tag)}{p(tag)} \\ &= \frac{I(first\_tag, tag)}{I(tag)} \end{aligned} \tag{5.1}$$

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 5.2:

$$\begin{aligned} p(first\_tag|tag) &= \frac{I(first\_tag, tag)}{I(tag)} * \sigma(I(first\_tag)) \\ &= \frac{record\_freq}{sum(record\_freq)} * \frac{1}{(1 + e^{-k * freq})} \end{aligned} \tag{5.2}$$

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 5.2 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*, $record\_freq = 10$, $sum(record\_freq) = 12$, $p(html) = 5,552$. 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 ques-

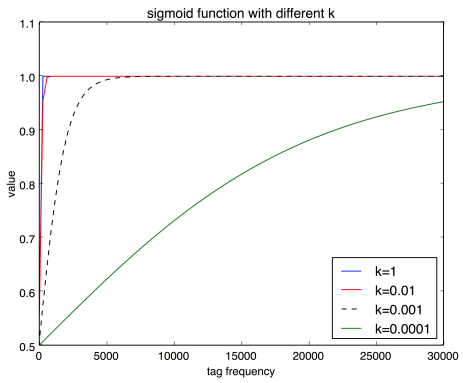


Figure 5.2: Shape of function $\frac{1}{(1+e^{-k*z})}$ for different values of k

tion. 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 5.3:

$$p_j = p_{1,j} + p_{2,j} * dis + ... + p_{k,j} * dis^{k-1} \tag{5.3}$$

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 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 5.1 presents the results of the enrichment of 8 tag lists (enriched tags are in bold).

Table 5.1: 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

5.1.3 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 (Mika 2007). Besides, (Yang 2013b) shows that tag frequency in Q&A sites also satisfie a power law distribution (Adamic 2000). 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 5.3 illustrates the process of building a tag tree. Figure 5.4 illustrates an example of *html*'s tree. Our topic extraction method is described in Algorithm 1.

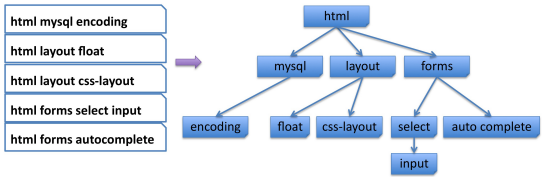


Figure 5.3: Example of a tag tree

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 Figure 5.1. 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,

Algorithm 1 Topic Extraction

Input: *enriched tag list of questions, topic number K* **Output:** *topic-tag distribution /*build trees process, shown in Fig 5.3*/* trees = null */* initialize */ tag* **in** *taglist* trees.insert(taglist) */*build affinity matrix for root_tags*/* root_tags = trees.get_root_tags() affinities_matrix = **build_affinity**(root_tags) */*run spectral-clustering on affinity matrix*/* groups = **spectral**(*affinities_matrix,K*) */*combine tree according to groups*/* new_trees = **combine_tree** (*trees,groups*) */*compute topic-tag distribution*/* topic_distributions = **compute** (*new_trees*) *** we perform a spectral clustering to divide these root tags into several groups*

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 5.4:

$$Simi(root\_i, root\_j) = \frac{I(root\_i, root\_j)}{(I(root\_i) + I(root\_j))} \tag{5.4}$$

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 (Ng 2001) on the affinity matrix to group these root nodes (line 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 (Chang 2013), 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 5.5:

$$p(tag|topic) = \frac{p(tag, topic)}{p(topic)} = \frac{I(tag) + 1}{I(sum(tag) + N)} \tag{5.5}$$

where $I(tag)$ denotes the number of occurrences of tag in the topic tree, and $I(sum(tag))$ 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-

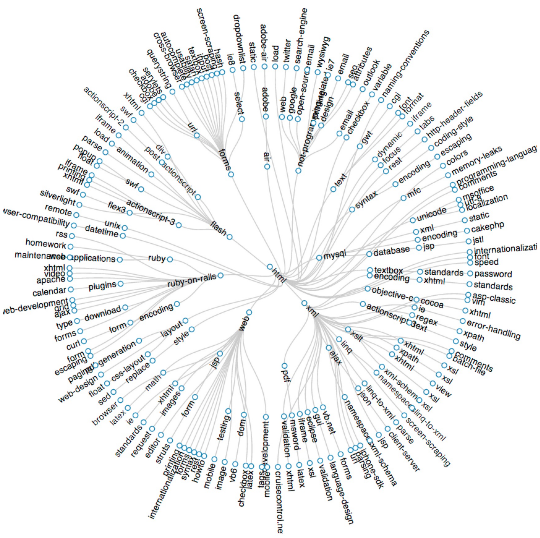


Figure 5.4: *html*'s tag tree

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 5.5. Table 5.2 shows the top tags and their probabilities detected by our method.

We used the spectral clustering implementation of scikit-learn toolkit[1]. We only run it on the root nodes, which have quite a small size (around 1175 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.

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

5.1.4 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 acquires the tags attached to this question and gradually, each user acquires a list of tags. So we represent a user by a tag list: $U = \{U_i | i = 1, ..., n\}, U_i = \{tag_i | i = m, n, ..., k\}$, and our goal is, for each user U_i, to find $I_i = \{I_{i1}, I_{i2}...I_{ik}\}$ 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 5.6 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 (Yang 2013a), 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.

$$I_{i,k} = log\left\{\sum_{t=1}^{v} P_{t,k} + 1\right\} \tag{5.6}$$

5.2 TTD Experiments and Evaluation on StackOverflow data

We conducted experiments on the dataset of activities on StackOverflow between 2008 and 2009, which is available [2], to evaluate the performance of our TTD ap-

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

proach 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.

5.2.1 Performance of Topic Extraction

We use the Perplexity (Blei 2003) 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 d^{th} question) the Perplexity score is computed as shown in equation 5.7:

$$Perplexity(D_{test}) = exp\left\{-\frac{\sum_{d=1}^{M}\log p(tag)}{\sum_{d=1}^{M} N_d}\right\} \tag{5.7}$$

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 5.1.4), 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 Figure 5.5. *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

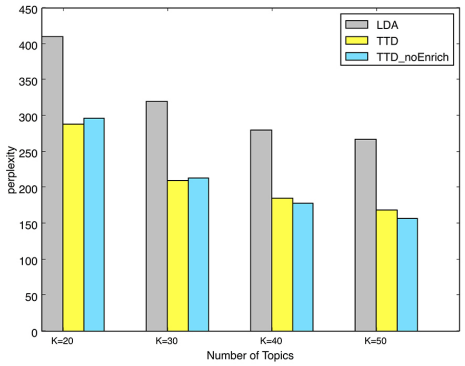


Figure 5.5: Comparison of topic extraction performances

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 5.3 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.

Table 5.2: Top tags and their probabilities for some topics computed with TTD

topic4		topic5		topic6	
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
topic12		**topic13**		**topic14**	
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

Table 5.3: 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

5.2.2 Performance of User Interest Detection

Traditional community detection algorithms are based on a network structure. As there is no explicit 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 (Gargi 2011). 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 $1 \sim 3$ 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 (Xie 2013): 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 (Yang 2013b), 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[3]. As 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 5.1.4) 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 5.8, 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 $1 \sim 3$ community labels for each user to indicate the level of interest. So we

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

evaluated each level of interest respectively.

$$M(C) = \frac{Avg\_inner(\sum_{i=1}^{N}\sum_{j=1}^{N} Simi(U\_i, U\_j))}{Avg\_rand(\sum_{i=1}^{N}\sum_{j=1}^{50} Simi(U\_i, Rand_{U}\_j))} \tag{5.8}$$

Experiment results are shown in Table 5.4. 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

Table 5.4: 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

5.5 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.

Table 5.5: 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) sql (21) tsql (6) performance (4) database (4) stored-procedures (3) sql-server-2005 (3) .net (3) mysql (2) sql-server-2000 (2)	java (135) swing (28) oracle (27) sql (23) subjective (15) windows (13) eclipse (12) best-practices (12) plsql (10) regex (10)	c# (107) jquery (89) javascript (56) .net (47) asp.net (27) css (23) regex (20) html (20) iphone (12) 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) javascript (193) mysql (116) html (86) css (57) regex (40) jquery (37) sql (27) ajax (26) apache (23)	c++ (703) c (187) templates (62) stl (53) linux (48) subjective (45) pointers (44) java (42) bash (40) boost (31)	cocoa (333) objective-c (184) iphone (47) cocoa-touch (39) osx (35) mac (34) iphone-sdk (20) xcode (18) cocoa-bindings (18) core-graphics (18)

5.2.3 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 at 100. We run the experiments on a computer with 3GHz Intel i7 CPU and 8GB RAM. From the experiment, we could find that LDA, SLPA and our method are linear in terms of the number of users. Although LDA algorithm is theoretically $O(nm)$ in each iteration, with n representing the number of users, and m representing the number of tags for each user, when we test it on large datasets, it clearly appears that only n actually has an impact; m has a very low impact. So LDA could be regarded as linear. Besides, (Griffiths 2004) proved that LDA model requires a few hundreds of iterations to obtain stable topic distribution. Our model does not have this limitation.

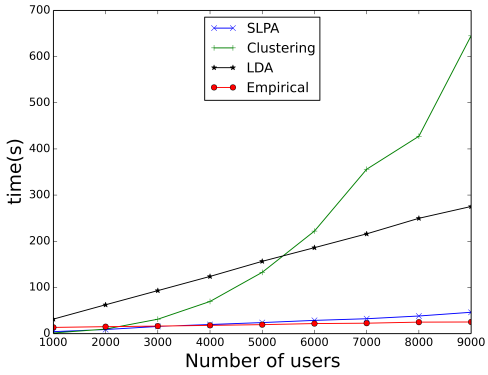


Figure 5.6: Scalability of the compared user interest detection methods

5.2.4 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. 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* (Leskovec 2008) with less *triangle-shape* structures. According to (Park 2013), the average number of *triangle-shape* structures per user in Twitter dataset is around 35714, 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 Figure 5.7, 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

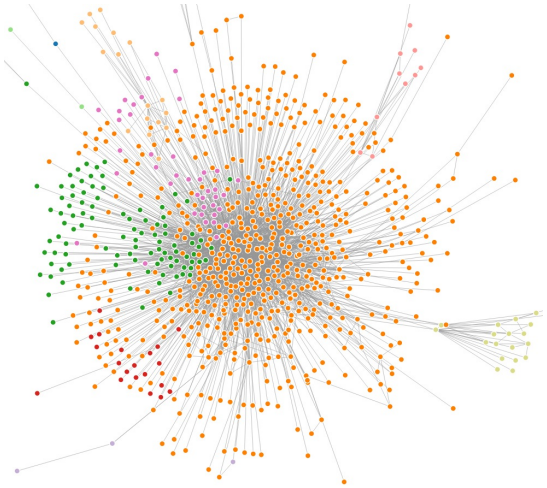


Figure 5.7: Illustration of co-answer-network

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 (Griffiths 2004) 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.

CHAPTER 6

Temporal Topic Expertise Activity (TTEA)

Contents

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*.

6.2 Solution

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 6.1.

Table 6.1: 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

6.2.1 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.

6.2.2 TTEA Model Structure

TTEA is an LDA-based model. Figure 6.1 represents it using the plate notation.

Let $u_i \in \{1, 2, ..., U\}$ be the set of users, $p_i \in \{1, 2, ..., P\}$ the set of answer

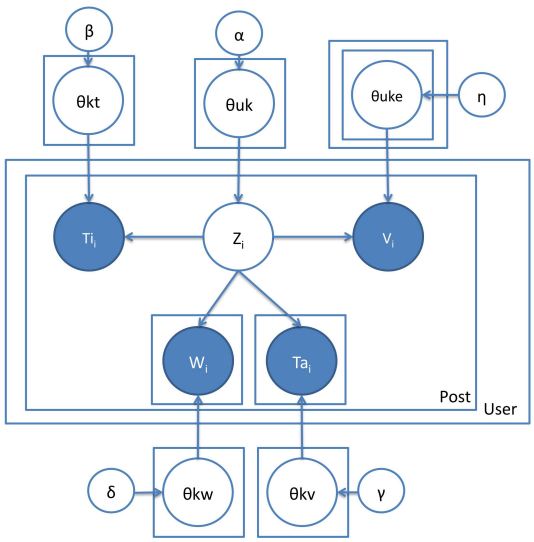


Figure 6.1: TTEA Model

posts, which are generated by these users, $w_i \in \{1, 2, ..., W\}$ the set of words in answers posts, $ta_i \in \{1, 2, ..., Ta\}$ the set of tags which are attached to posts, $v_i \in \{1, 2, ..., V\}$ the set of votes for each answer posts, $ti_i \in \{1, 2, ..., Ti\}$ the set of points in time which could be months or days depending on the requirements, and $z_i \in \{1, 2, ..., K\}$ 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 6.1.

Contrary to (Blei 2003) 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.

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 3000. Since

the vote scores' counts distribution follows a log distribution(Yang 2013b), 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 (Wang 2006) (Hu 2014), 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 u-th user, $u \in U$

  -draw user topic distribution $\theta_{uk} \sim$ Dir(α)

- For the k-th topic, $k \in K$

  -draw topic tag distribution $\theta_{kv} \sim$ Dir(γ)

  -draw topic word distribution $\theta_{kw} \sim$ Dir(δ)

  -draw topic time distribution $\theta_{kt} \sim$ Dir(β)

- For the u-th user, $u \in U$

  -for the k-th topic, $k \in K$

  —draw user topic expertise distribution $\theta_{uke} \sim$ Dir(η)

- For the u-th user, $u \in U$

-for the n-th q&a post, $p \in P$

—draw topic $z \sim$ Multi(θ_{uk})

—draw time point $t \sim$ Multi(θ_{kt})

—for the i-th word, $w \in W$

——draw word $w \sim$ Multi(θ_{kw})

—for the j-th tag, $ta \in Ta$

——draw tag $t \sim$ Multi(θ_{kv})

—draw expertise level $v \sim$ Multi(θ_{uke})

6.2.3 TTEA Model Inference

Like (Hu 2014), we use the collapsed Gibbs Sampling algorithm (Griffiths 2004) to sample the hidden variable z, based on which the unknown probabilities $\{\theta_{uk}, \theta_{kv}, \theta_{kw}, \theta_{kt}$, and θ_{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 6.1. As explained before, each answer post will have one topic assignment.

$$
\begin{aligned}
& p(z_i = k | z_{\neg i}, \mathbf{U}, \mathbf{Ti}, \mathbf{Ta}, \mathbf{W}) \\
& \propto \frac{C^k_{u,\neg i} + \alpha_1}{\sum_{k=1}^{K} C^k_{u,\neg i} + K * \alpha_1} \\
& \cdot \frac{\prod_{ta=1}^{Ta} \prod_{q=0}^{C_{ta}-1} (C^{ta}_{k,\neg i} + q + \gamma)}{\prod_{p=0}^{\sum C_{ta}-1} \sum_{ta=1}^{Ta} (C^{v}_{k,\neg i} + p + Ta * \gamma)} \\
& \cdot \frac{\prod_{w=1}^{W} \prod_{s=0}^{C_w-1} (C^{w}_{k,\neg i} + s + \delta)}{\prod_{t=0}^{\sum C_w-1} \sum_{w=1}^{W} (C^{w}_{k,\neg i} + t + W * \delta)} \\
& \cdot \frac{C^{ti}_{k,\neg i} + \beta}{\sum_{ti=1}^{Ti} C^{ti}_{k,\neg i} + Ti * \beta} \\
& \cdot \frac{C^{e}_{u,k,\neg i} + \eta}{\sum_{e=1}^{E} C^{e}_{u,k,\neg i} + E * \eta}
\end{aligned}
\tag{6.1}
$$

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:

$$\theta_{uk} = \frac{C^k_u + \alpha_1}{\sum_{k=1}^{K} C^k_u + K * \alpha_1} \tag{6.2}$$

$$\theta_{kv} = \frac{C^{ta}_k + \gamma}{\sum_{ta=1}^{Ta} C^{ta}_k + Ta * \gamma} \tag{6.3}$$

$$\theta_{kw} = \frac{C^w_k + \delta}{\sum_{w=1}^{W} C^w_k + W * \delta} \tag{6.4}$$

$$\theta_{kt} = \frac{C^{ti}_k + \beta_2}{\sum_{ti=1}^{Ti} C^{ti}_k + Ti * \beta_2} \tag{6.5}$$

$$\theta_{uke} = \frac{C^e_{u,k} + \eta}{\sum_{e=1}^{E} C^e_{u,k} + E * \eta} \tag{6.6}$$

6.2.4 Post Processing

The above model can only generate the distributions {θ_{uk}, θ_{kv}, θ_{kw}, θ_{kt}, and θ_{uke} }. To generate the other distributions, e.g. θ_{ku}, θ_{tk} and $_{ukt}$, 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 to topic k and posted at time ti. Then, we estimate θ_{ku}, θ_{tk}, $_{ukt}$ according to the following equations:

$$\theta_{ku} = \frac{C^u_k + \alpha_2}{\sum_{u=1}^{U} C^u_k + U * \alpha_2} \tag{6.7}$$

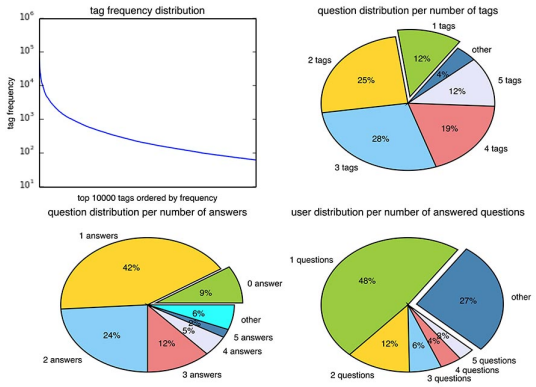


Figure 6.2: Basic perspectives of the dataset

$$\theta_{tk} = \frac{C_{ti}^{k} + \beta_1}{\sum_{k=1}^{K} C_{ti}^{k} + K * \beta_1} \tag{6.8}$$

$$\theta_{ukt} = \frac{C_{u,k}^{ti} + \lambda}{\sum_{ti=1}^{T} C_{u,k}^{ti} + T * \lambda} \tag{6.9}$$

6.3 TTEA Experiments and Evaluation on StackOverflow data

6.3.1 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 6.2 and figure 6.2 provide basic statistics on the dataset.

Table 6.2: 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

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.

6.3.2 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: (Yang 2013b) 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: (Guo 2008b) 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: (Hu 2014) 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 gen-

erate user-topic distribution, so we compute it with the user-group distribution and group-topic distribution.

– LDA: based on (Blei 2003) 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 (Chang 2013) and the same number of expertises E=*10* as (Yang 2013b), which have proved to be a reasonable setting for the Stackoverflow dataset. We empirical set Dirichlet hyper parameters α_1=α_2=50/K, β_1=β_2=0.01, δ=λ=η=0.01, γ=0.001 according to suggestions in (Griffiths 2004).

6.3.3 Performance of Topic Extraction

Table 7.2 and Table 6.4 show the top tags and words detected by our model. We use again the Perplexity (Blei 2003) 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 1^{st} 2011 to October 1^{st} 2011, from users having more than 80 posts (as in (Yang 2013b)). The resulting training dataset contains 87516 q&a posts by 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 1^{th} 2011 to January 1^{th} 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 i^{th} post and the Perplexity score is computed according to equation 6.10.

$$Perplexity(D_{test}) = exp\left\{-\frac{\sum_{i=1}^{M}\log p(W_i)}{\sum_{i=1}^{M} N_i}\right\} \tag{6.10}$$

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 6.11.:

$$P(W_i) = \sum_k \theta_{u_i k} \prod_w \theta_{k w_i} \tag{6.11}$$

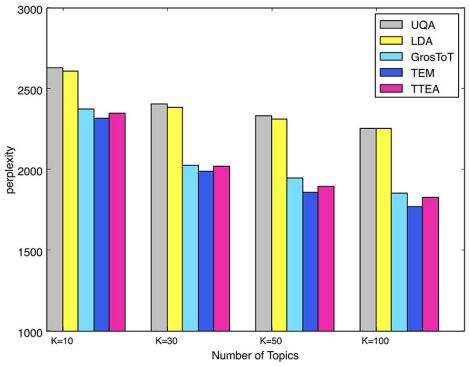


Figure 6.3: Comparison of topic extraction performances

Table 6.3: 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

Figure 6.3 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.

(Chang 2009) suggested that topic models should focus on evaluations on real-world

Table 6.4: 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

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.

6.3.4 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 ($Sim(u, q)$). The intuition behind equation 6.12 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.

$$Sim(u, q) = (1 - JS(\theta_{uk}, \theta_{qk})) \tag{6.12}$$

where θ_{uk} is the user topic interest distribution, θ_{qk} is the question topic distribution, and JS(.) is the JensenâShannon divergence distance. We obtain θ_{uk} directly from model results. For θ_{qk}, we apply equation 6.13.

$$\begin{aligned} \theta q, k &\propto p(k|w_q, t_q, u) \\ &= p(k|u)p(w_q|k)p(t_q|k) \\ &= \theta uk \sum_{w_i \in w_q} \theta_{kw_i} \sum_{t_i \in t_q} \theta_{kv_i} \end{aligned} \tag{6.13}$$

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 1^{th} 2011 to October 1^{th} 2011 from users having more than 50 q&a posts for the training dataset. Rather than using thethreshold of 80 post like in (Yang 2013b), we empirically set it to 50 posts to get enough users for recommendation. The resulting training set contains 297881 posts by 2555 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 1^{th} 2011 to January 1^{th} 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 6044 questions, 18077 answers and 7888 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 6.3.5.

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 6.14 and Recal@N is defined in equation 6.15.

$$P@N = \frac{1}{|Q|} \sum_{q \in Q} \frac{|R_q \cap U_q|}{|R_q|} \tag{6.14}$$

$$R@N = \frac{1}{|Q|} \sum_{q \in Q} \frac{|R_q \cap U_q|}{|U_q|} \tag{6.15}$$

where Q is the set of testing questions. Like in (Chang 2013), we use the Matching Set Count (MSC) which is defined in equation 6.16. The idea is to count the number of successful recommendations, i.e., for which at least one of the recommended users answered the question.

$$MSC@N = \frac{1}{|Q|} \sum_{q \in Q} 1[R_q \cap U_q \neq \emptyset] \tag{6.16}$$

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

Table 6.5: 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 @2
TTEA	0.024	0.019	0.015	0.013	0.045	0.072	0.111	0.142	0.112	0.178	0.269
TTEA-ACT	0.028	**0.022**	**0.017**	**0.014**	0.052	**0.083**	**0.127**	**0.159**	0.134	**0.209**	**0.313**
TEM	0.024	0.019	0.015	0.013	0.045	0.073	0.114	0.146	0.114	0.179	0.275
TEM-ACT	0.029	**0.023**	**0.018**	**0.015**	0.054	**0.084**	**0.129**	**0.162**	0.137	**0.210**	**0.315**
UQA	**0.030**	0.019	0.012	0.010	**0.062**	0.075	0.095	0.112	**0.149**	0.179	0.224
GROSTOT	0.027	0.017	0.011	0.009	0.055	0.067	0.085	0.099	0.134	0.164	0.204
RANDOM	0.001	0.001	0.001	0.001	0.001	0.002	0.005	0.007	0.003	0.007	0.013

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 6.17, where $Act(u, q)$ 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.

$$\begin{aligned} SimAct(u,q) &= (1 - JS(\theta_{uk}, \theta_{qk})) * Act(u,q) \\ &= (1 - JS(\theta_{uk}, \theta_{qk})) * \sum_{k=1}^{K} \theta_{qk} * \theta_{ku} \end{aligned} \tag{6.17}$$

Table 6.5 shows the results. We ran the experiments five times and listed the average scores. Our observations can be summerized 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 if TEM or TEM-ACT perform better than our model they remain again time consuming. Experiments show that the training process takes around 3$\sim$4 times longer compared to our model.

6.3.5 Experiment Parameter Sensitivity Analysis

For the training dataset, we used all the posts in a three months period, from January 1^{th} 2011 to March 31^{th} 2011, from users having at least 50 q&a posts, rather than 80 posts like (Yang 2013b), in order to get enough users for recommendations. The training set contains 371181 posts by 3123 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 1^{th} 2011 to June 31^{th} 2011. Therefore the training and testing datasets have no overlaps. We removed questions with no or only one answer. The resulting test dataset contains 9048 questions, 27870 answers and 10147 users. Table 6.6 shows the question routing results. We can still find that TTEA-ACT outperforms all the

Table 6.6: 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 @2
TTEA	0.026	0.020	0.015	0.013	0.047	0.073	0.110	0.136	0.123	0.186	0.273
TTEA-ACT	**0.032**	**0.026**	**0.019**	**0.016**	**0.058**	**0.093**	**0.137**	**0.168**	**0.153**	**0.236**	**0.339**
TEM	0.025	0.021	0.016	0.013	0.047	0.076	0.112	0.139	0.120	0.191	0.274
TEM-ACT	**0.032**	**0.025**	**0.020**	**0.016**	**0.058**	**0.092**	**0.141**	**0.171**	**0.153**	**0.235**	**0.348**
UQA	0.027	0.016	0.011	0.009	0.052	0.062	0.080	0.096	0.130	0.155	0.196
GROSTOT	0.023	0.014	0.009	0.007	0.044	0.055	0.069	0.081	0.112	0.137	0.172
RANDOM	0.001	0.001	0.001	0.001	0.001	0.002	0.004	0.005	0.003	0.005	0.010

Table 6.7: 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 @2
TTEA	0.016	0.013	0.012	0.010	0.030	0.050	0.086	0.112	0.076	0.127	0.213
TTEA-ACT	0.023	**0.018**	**0.015**	**0.012**	0.042	**0.066**	**0.107**	**0.134**	**0.112**	**0.170**	**0.268**
TEM	0.017	0.015	0.012	0.010	0.032	0.054	0.091	0.115	0.083	0.137	0.222
TEM-ACT	0.024	**0.018**	**0.014**	**0.012**	0.043	**0.068**	**0.103**	**0.131**	**0.114**	**0.172**	**0.254**
UQA	**0.028**	0.016	0.011	0.008	**0.056**	**0.066**	0.083	0.099	0.137	0.159	0.199
Grostot	0.023	0.015	0.010	0.008	0.045	0.058	0.075	0.089	0.112	0.143	0.183
Random	0.001	0.001	0.001	0.001	0.002	0.003	0.004	0.006	0.005	0.008	0.012

baseline models. Besides, Both TTEA-ACT and TEM-ACT outperform all the other models.

Table 6.7 shows the question routing results with a number of topics set to 15. We use the same training and testing datasets as in section 6.3.4.

Table 6.8 shows the question routing results for the number of topics set to 50. We use the same training and testing datasets as in section 6.3.4.

Table 6.9 shows the question routing results whith users having more than 40 posts.

Table 6.8: 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 @2
TTEA	0.028	0.023	0.018	0.015	0.054	0.087	0.132	0.168	0.134	0.215	0.319
TTEA-ACT	**0.033**	**0.025**	**0.019**	**0.016**	0.063	**0.095**	**0.142**	**0.178**	**0.158**	**0.235**	**0.343**
TEM	0.029	0.024	0.018	0.015	0.056	0.088	0.136	0.171	0.141	0.220	0.325
TEM-ACT	**0.033**	**0.026**	**0.020**	**0.017**	0.062	**0.096**	**0.145**	**0.182**	**0.157**	**0.240**	**0.347**
UQA	0.032	0.019	0.012	0.010	**0.065**	0.077	0.097	0.116	**0.158**	0.185	0.227
Grostot	0.028	0.017	0.011	0.009	0.056	0.067	0.088	0.102	0.136	0.163	0.210
Random	0.001	0.001	0.001	0.001	0.002	0.002	0.005	0.007	0.004	0.006	0.013

Table 6.9: 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
TTEA	0.021	0.018	0.014	0.012	0.040	0.067	0.104	0.132	0.100	0.167	0.2
TTEA-ACT	0.026	**0.021**	**0.016**	**0.014**	0.049	**0.076**	**0.118**	**0.149**	0.126	**0.193**	**0.2**
TEM	0.023	0.018	0.014	0.012	0.043	0.069	0.106	0.137	0.109	0.170	0.2
TEM-ACT	0.027	**0.021**	**0.016**	**0.014**	0.050	**0.078**	**0.121**	**0.152**	0.128	**0.194**	**0.2**
UQA	**0.029**	0.018	0.011	0.009	**0.059**	0.071	0.087	0.101	**0.142**	0.169	0.2
Grostot	0.025	0.016	0.010	0.008	0.050	0.063	0.077	0.091	0.122	0.152	0.1
Random	0.000	0.000	0.000	0.000	0.001	0.002	0.003	0.005	0.002	0.004	0.0

Table 6.10: 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
TTEA	0.028	0.023	0.019	0.016	0.051	0.083	0.135	0.175	0.132	0.212	0.3
TTEA-ACT	0.031	**0.026**	**0.020**	**0.018**	0.058	**0.094**	**0.146**	**0.188**	0.150	**0.238**	**0.3**
TEM	0.031	0.026	0.020	0.017	0.056	0.095	0.147	0.188	0.143	0.238	0.3
TEM-ACT	0.035	**0.027**	**0.021**	**0.018**	0.063	**0.100**	**0.151**	**0.193**	0.165	**0.253**	**0.3**
UQA	**0.040**	0.025	0.016	0.013	**0.077**	0.096	0.124	0.150	**0.194**	0.237	0.2
Grostot	0.036	0.022	0.015	0.012	0.070	0.086	0.114	0.135	0.177	0.214	0.2
Random	0.001	0.001	0.001	0.001	0.002	0.003	0.006	0.011	0.005	0.008	0.0

We use the same period of dataset used in section 6.3.4. Due to the different filter limit, the training set contains 3457 users and 338485 q&a posts, the testing set contains 8579 questions, 25500 answers and 10135 involved users.

Table 6.10 shows the question routing results with users having more than 80 posts. We use the same period of dataset used in section 6.3.4. Due to the different filter limit, the training set contains 1275 users and 216940 q&a posts, the testing set contains 2589 questions, 8006 answers and 4196 involved users.

6.3.6 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 $SimExp(u, q)$ between user topic interest and user topic expertise to answer question q. The intuition behind equation 6.18 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.

$$SimExp(u,q) = (1 - JS(\theta_{uk}, \theta_{qk})) * Exp(u,q) \tag{6.18}$$

where θ_{uk}, θ_{qk} is the same than in 6.12 for user topic interest distribution. For our method, we compute $Exp(u,q)$ by equation 6.19

$$Exp(u,q) = \sum_{e=1}^{E} \theta_{kue} * e \tag{6.19}$$

As UQA and GROSTOT do not model expertise, like (Yang 2013b), we set $Exp(u,q)$ to 1 for these two methods. For TEM, we reuse equation 6.20 indicated in (Yang 2013b).

$$Exp(u,q) = \sum_{e=1}^{E} \phi_{z,u,e} * \mu_e \tag{6.20}$$

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 N-th user, recommended by an algorithm, not only answers the question but also gets the highest votes.

Table 6.11: 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

Table 6.11 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.

6.3.7 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 6.4-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 6.4-b shows the dynamics of $Android$, $Iphone$ and $Flash$ related topics from July 1^{st} 2011 to July 31^{st} 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 6.4-c shows the dynamicsof $Android$, $Iphone$ 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

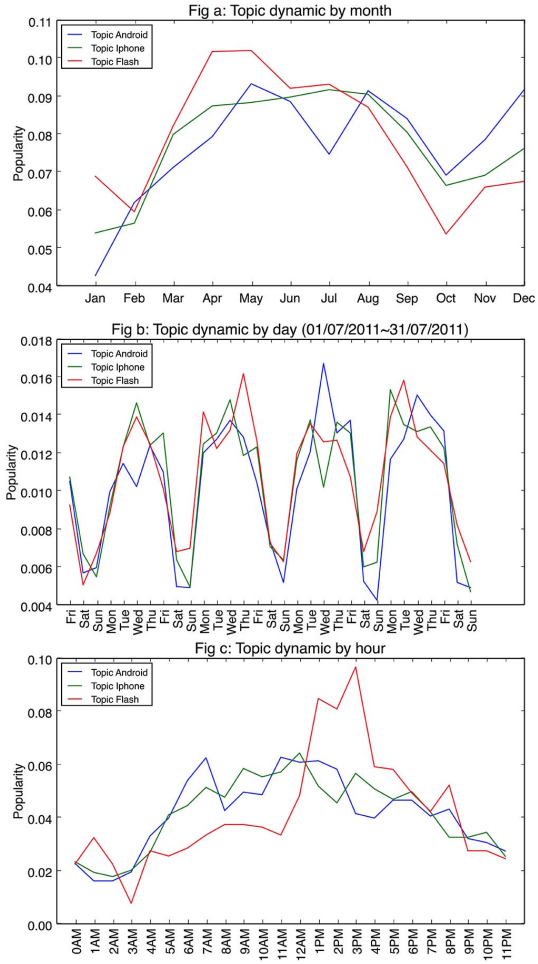


Figure 6.4: Topic dynamics

of θ_{ku} in $Android$ related topic and $Iphone$ related topic separately. Figure 6.5-a,b show the activity pattern of the two most active users in $Iphone$ related topic. We can observe that the user in Figure 6.5-a is only active during work-time. The user seldom answers questions after 7PM. On the contrary, the user in Figure 6.5-b is active until very late but not midnight. Figure 6.5-c,d show the activity pattern of the two most active users in $Android$ related topic. We can observe that the user in Figure 6.5-c is active in the morning, afternoon and evening. On the contrary, the user in Figure 6.5-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.

Figure 6.5: User Topic Activities

CHAPTER 7

Automatically generate a label for a bag of words

Contents

7.1 Introduction

The outputs of topic model are normally bag of words. Each of them represent a topic. A obvious question will be how to give a general label or name for these bag of words. It can help to management the topic model result.

7.2 Solution

Our previous work on topic modeling can generate topics based on question answer contents and tags. Each topic is consist of several tags or words. Table 7.2 list some detected topics from a Flickr dataset and StackOverflow dataset. The topic extraction

algorithm are able to put closely related words or tags into the same topic, however, it can not generate a label to describe the meaning of the bag of words.

Table 7.1: Top tags and their probabilities on flickr dataset

topic3		topic4		topic5	
airplane	0.074	tshirt	0.216	music	0.077
airport	0.053	shirt	0.154	rock	0.040
aircraft	0.029	shirts	0.112	concert	0.036
flying	0.028	threadless	0.109	live	0.025
plane	0.027	tshirts	0.009	band	0.022
aviation	0.022	tee	0.008	singing	0.019
flight	0.014	clothing	0.007	guitar	0.018
aeroplane	0.012	media	0.006	festival	0.017
jet	0.010	models	0.006	show	0.014
boeing	0.009	camiseta	0.004	livemusic	0.010
topic23		topic24		topic25	
italy	0.179	bike	0.114	portrait	0.049
italia	0.053	motorcycle	0.052	girl	0.029
rome	0.028	racing	0.033	woman	0.014
florence	0.021	bicycle	0.028	smile	0.014
venice	0.014	race	0.027	model	0.010
tuscany	0.014	motorbike	0.024	sexy	0.009
roma	0.011	sport	0.019	face	0.008
europe	0.011	speedway	0.011	fun	0.008
firenze	0.010	500cc	0.010	man	0.008
milan	0.007	methanol	0.010	love	0.008

7.2.1 link to dbpedia

DBpedia is a crowd-soured community effort to extract structured information from Wikipedia and make this information available on the Web. In order to use these extra information, a basic step is linking those words to dbpedia.

However, in some cases, several resources corresponding to the same word or tag. for instance, *java* could link to *Java* as an island, it could also link to *Java* as java programming language. Therefore, we have to deal with disambiguation problem when linking tag to dbpedia resource. There are work on named entity detection and entity linking. So we directly reuse these work.

7.2.2 Another way to disambiguation

As our data source is from StackOverflow website, we found there are detailed descriptions for each tag on the website. We also found that each resource in DBpedia

Table 7.2: Top tags and their probabilities on stackoverflow dataset

topic4		topic5		topic6	
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
topic12		topic13		topic14	
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

has descriptions. The idea is to compute the cosine similarity between the two description to solve the disambiguation problem. The process is describe as following.

We use dbpedia keyword lookup service to retrieve related resource for each tag. Results of this service is a lits of resources related to the given tag.

1> retrieve tag description from StackOverflow. 2> using dbpedia lookup service to get results for the tag 3> for each result compute the cosine distances between the two descriptions. 4> choose the highest one to link to dbpedia.

After linking all the tags to the corresponding resources. We then perform a query to retrieve the potential relation graph among those tags in a topic. Then we run several graph algorithm on the detected graph to chose a label to annotate the bag of words.

7.2.2.1 A naive approach

As the topic modeling algorithm normal generate a topic-word distribution to indicate to what extent a word is related to a topic. By sorting words' corresponding probabilities, we can obtain a ranked word list for each topic, which are the top related words in each topic. A naive approach would be using the first one or two

words to label each topic. It does not need any extra knowledges.

7.2.3 Results

In order to evaluate the performance of different ways to generate this label. we conduct user studies on the results.

CHAPTER 8

Conclusion

In this thesis, 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. 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 experiments 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 illus-

trated 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.

APPENDIX A

Appendix Example

A.1 Appendix Example section

And I cite myself to show by bibtex style file (two authors) (**?**).

This for other bibtex stye file : only one author (**?**) and many authors (**?**).

Bibliography

[Adamic 2000] Lada A Adamic and Bernardo A Huberman. *Power-law distribution of the world wide web*. Science, vol. 287, no. 5461, pages 2115–2115, 2000. (Cited on page 49.)

[Ahn 2010] Yong-Yeol Ahn, James P Bagrow and Sune Lehmann. *Link communities reveal multiscale complexity in networks*. Nature, vol. 466, no. 7307, pages 761–764, 2010. (Cited on page 21.)

[Anderson 2012] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg and Jure 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, pages 850–858. ACM, 2012. (Cited on pages 3, 14, 24 and 31.)

[Balasubramaniam 2009] Niroshan Balasubramaniam. *User-generated content*. In Proceedings of business aspects of the internet of things, seminar of advanced topics, pages 28–33. ETH Zurich, 2009. (Cited on page 12.)

[Berners-Lee 2001] Tim Berners-Lee, James Hendler, Ora Lassila*et al. The semantic web*. Scientific american, vol. 284, no. 5, pages 28–37, 2001. (Cited on pages 14 and 16.)

[Berners-Lee 2006] Tim Berners-Lee. *Linked data-design issues*. 2006. (Cited on page 16.)

[Bizer 2011] Christian Bizer. *Evolving the Web into a Global Data Space.* In BNCOD, volume 7051, page 1, 2011. (Cited on page 16.)

[Blei 2003] David M Blei, Andrew Y Ng and Michael I Jordan. *Latent dirichlet allocation*. the Journal of machine Learning research, vol. 3, pages 993–1022, 2003. (Cited on pages 37, 53, 64 and 70.)

[Bouguessa 2008] Mohamed Bouguessa, Benoît Dumoulin and Shengrui 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, pages 866–874. ACM, 2008. (Cited on page 24.)

[Chang 2009] Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish and David M. Blei. *Reading Tea Leaves: How Humans Interpret Topic Models*. In Neural Information Processing Systems, 2009. (Cited on page 71.)

[Chang 2013] Shuo Chang and Aditya Pal. *Routing Questions for Collaborative Answering in Community Question Answering*. In Proceedings of the 2013 IEEE/ACM ASONAM, pages 494–501, New York, NY, USA, 2013. ACM. (Cited on pages 22, 24, 50, 70 and 74.)

[DiNucci 2012] Darcy DiNucci. *âFragmented Futureâ, 1999*. Dostupn é z: http://www. darcyd. com/fragmented future. pdf, 2012. (Cited on page 11.)

[Feigenbaum 2007] Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann and Susie Stephens. *The semantic web in action*. Scientific American, vol. 297, no. 6, pages 90–97, 2007. (Cited on page 16.)

[Gargi 2011] Ullas Gargi, Wenjun Lu, Vahab S. Mirrokni and Sangho Yoon. *Large-Scale Community Detection on YouTube for Topic Discovery and Exploration*. In ICWSM, 2011. (Cited on pages 21 and 55.)

[Griffiths 2004] Thomas L Griffiths and Mark Steyvers. *Finding scientific topics*. Proceedings of the National Academy of Sciences, vol. 101, no. suppl 1, pages 5228–5235, 2004. (Cited on pages 39, 58, 60, 66 and 70.)

[Guo 2008a] Jinwen Guo, Shengliang Xu, Shenghua Bao and Yong Yu. *Tapping on the Potential of Q&a Community by Recommending Answer Providers*. In Proceedings of the 17th ACM Conference on Information and Knowledge

Management, CIKM '08, pages 921–930, New York, NY, USA, 2008. ACM. (Cited on pages 27 and 29.)

[Guo 2008b] Jinwen Guo, Shengliang Xu, Shenghua Bao and Yong Yu. *Tapping on the potential of q&a community by recommending answer providers*. In Proceedings of the 17th ACM CIKM, pages 921–930. ACM, 2008. (Cited on pages 24 and 69.)

[Harper 2008] F. Maxwell Harper, Daphne Raban, Sheizaf Rafaeli and Joseph A. Konstan. *Predictors of Answer Quality in Online Q&A Sites*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08, pages 865–874, New York, NY, USA, 2008. ACM. (Cited on page 13.)

[Hofmann 1999] Thomas Hofmann. *Probabilistic latent semantic analysis*. In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pages 289–296. Morgan Kaufmann Publishers Inc., 1999. (Cited on page 25.)

[Hu 2014] Zhiting Hu, Junjie Yao and Bin Cui. *User Group Oriented Temporal Dynamics Exploration*. In Twenty-Eighth AAAI14, 2014. (Cited on pages 25, 65, 66 and 69.)

[Ji 2013] Zongcheng Ji and Bin Wang. *Learning to rank for question routing in community question answering*. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pages 2363–2368. ACM, 2013. (Cited on page 24.)

[Jurczyk 2007] Pawel Jurczyk and Eugene Agichtein. *Discovering authorities in question answer communities by using link analysis*. In Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, pages 919–922. ACM, 2007. (Cited on page 24.)

[Lancichinetti 2011] Andrea Lancichinetti, Filippo Radicchi, José J Ramasco and

Santo Fortunato. *Finding statistically significant communities in networks*. PloS one, vol. 6, no. 4, page e18961, 2011. (Cited on page 22.)

[Leskovec 2008] Jure Leskovec, Kevin J Lang, Anirban Dasgupta and Michael W Mahoney. *Statistical properties of community structure in large social and information networks*. In Proceedings of the 17th international conference on World Wide Web, pages 695–704. ACM, 2008. (Cited on page 59.)

[Li 2010a] Baichuan Li and Irwin King. *Routing questions to appropriate answerers in community question answering services*. In Proceedings of the 19th ACM international conference on Information and knowledge management, pages 1585–1588. ACM, 2010. (Cited on pages 3 and 24.)

[Li 2010b] Daifeng Li, Bing He, Ying Ding, Jie Tang, Cassidy Sugimoto, Zheng Qin, Erjia Yan, Juanzi Li and Tianxi Dong. *Community-based Topic Modeling for Social Tagging*. In Proc. of the 19th ACM CIKM, CIKM '10, pages 1565–1568, New York, NY, USA, 2010. ACM. (Cited on page 38.)

[Liu 2009] Ling Liu and M Tamer Zsu. Encyclopedia of database systems. Springer Publishing Company, Incorporated, 2009. (Cited on page 19.)

[Ma 2015] Zongyang Ma, Aixin Sun, Quan Yuan and Gao Cong. *A Tri-Role Topic Model for Domain-Specific Question Answering*. In Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015. (Cited on page 24.)

[McDaid 2010] Aaron McDaid and Neil Hurley. *Detecting highly overlapping communities with model-based overlapping seed expansion*. In Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, pages 112–119. IEEE, 2010. (Cited on page 22.)

[Mika 2007] Peter Mika. *Ontologies are us: A unified model of social networks and semantics*. Web Semantics: Science, Services and Agents on the World Wide Web, vol. 5, no. 1, pages 5–15, 2007. (Cited on page 49.)

[Moens 2014] Marie-Francine Moens, Juanzi Li and Tat-Seng Chua. Mining user generated content. CRC Press, 2014. (Cited on pages 1 and 12.)

[Ng 2001] Andrew Y. Ng, Michael I. Jordan and Yair Weiss. *On Spectral Clustering: Analysis and an algorithm*. In ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, pages 849–856. MIT Press, 2001. (Cited on page 50.)

[O'really 2009] Tim O'really. *Design Patterns and Business Models for the Next Generation of Software*. URL: http://oreilly. com/web2/archive/what-is-web-20. html (01.12. 2013.), 2009. (Cited on page 11.)

[Pal 2011] Aditya Pal, Rosta Farzan, Joseph A Konstan and Robert E Kraut. *Early detection of potential experts in question answering communities*. In User Modeling, Adaption and Personalization, pages 231–242. Springer, 2011. (Cited on page 24.)

[Park 2013] Ha-Myung Park and Chin-Wan Chung. *An efficient MapReduce algorithm for counting triangles in a very large graph*. In Proceedings of the 22nd ACM CIKM13, pages 539–548. ACM, 2013. (Cited on page 59.)

[Passant 2009] Alexandre Passant. *Technologies du Web Sémantique pour lâEntreprise 2.0*. PhD thesis, PhD thesis, Université Paris IV-Sorbonne, 2009. (Cited on page 11.)

[Porter 2010] Joshua Porter. Designing for the social web, ebook. Peachpit Press, 2010. (Cited on page 10.)

[Rheingold 2000] Howard Rheingold. The virtual community: Homesteading on the electronic frontier. MIT press, 2000. (Cited on page 10.)

[Sang-Hun 2007] CHOE Sang-Hun. *South Koreans Connect Through Search Engine*. New York Times, 2007. (Cited on page 13.)

[Shah 2010] Chirag Shah and Jefferey Pomerantz. *Evaluating and Predicting Answer Quality in Community QA*. In Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10, pages 411–418, New York, NY, USA, 2010. ACM. (Cited on page 14.)

[Smith 2012] Andrew N Smith, Eileen Fischer and Chen Yongjian. *How does brand-related user-generated content differ across YouTube, Facebook, and Twitter?* Journal of Interactive Marketing, vol. 26, no. 2, pages 102–113, 2012. (Cited on page 12.)

[Sun 2013] Xiaoling Sun and Hongfei Lin. *Topical community detection from mining user tagging behavior and interest.* JASIST, vol. 64, no. 2, pages 321–333, 2013. (Cited on page 22.)

[Tang 2008] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang and Zhong Su. *ArnetMiner: extraction and mining of academic social networks*. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 990–998. ACM, 2008. (Cited on page 22.)

[Wang 2006] Xuerui Wang and Andrew 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, pages 424–433. ACM, 2006. (Cited on pages 25 and 65.)

[Watson 2008] Tom Watson. Causewired: plugging in, getting involved, changing the world. John Wiley & Sons, 2008. (Cited on page 10.)

[Web 2007] Participative Web and User-Created Content. *Web 2.0, Wikis and Social Networking*. SourceOCDE Science et technologies de l'information, vol. 15, 2007. (Cited on page 12.)

[Xie 2013] Jierui Xie, Stephen Kelley and Boleslaw K. Szymanski. *Overlapping*

community detection in networks: The state-of-the-art and comparative study. ACM Comput. Surv., vol. 45, no. 4, page 43, 2013. (Cited on pages 21 and 55.)

[Xu 2012] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng and James Cheng. *A model-based approach to attributed graph clustering*. In SIGMOD Conference, pages 505–516, 2012. (Cited on page 21.)

[Yang 2013a] Jaewon Yang, Julian McAuley and Jure Leskovec. *Community detection in networks with node attributes*. In Data Mining (ICDM), 2013 IEEE 13th International Conference on, pages 1151–1156. IEEE, 2013. (Cited on pages 22, 23 and 52.)

[Yang 2013b] Liu Yang, Minghui Qiu, Swapna Gottipati, Feida Zhu, Jing Jiang, Huiping Sun and Zhong Chen. *CQArank: jointly model topics and expertise in community question answering*. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pages 99–108. ACM, 2013. (Cited on pages 24, 31, 49, 56, 65, 69, 70, 73, 75 and 78.)

[Yin 2013] Hongzhi Yin, Bin Cui, Hua Lu, Yuxin Huang and Junjie Yao. *A unified model for stable and temporal topic detection from social media data*. In Data Engineering (ICDE), 2013 IEEE 29th International Conference on, pages 661–672. IEEE, 2013. (Cited on page 25.)

[Zhang 2007a] Haizheng Zhang, Baojun Qiu, C. Lee Giles, Henry C. Foley and John Yen. *An LDA-based Community Structure Discovery Approach for Large-Scale Social Networks*. In ISI, pages 200–207, 2007. (Cited on page 22.)

[Zhang 2007b] Jun Zhang, Mark S Ackerman and Lada Adamic. *Expertise networks in online communities: structure and algorithms*. In Proceedings of the 16th

international conference on World Wide Web, pages 221–230. ACM, 2007. (Cited on page 24.)

[Zhou 2012] Tom Chao Zhou, Michael R. Lyu and Irwin 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, pages 783–790, New York, NY, USA, 2012. ACM. (Cited on page 3.)

Temporal and semantic analysis of richly typed social networks from user-generated content sites on the web

Abstract:

Keywords: Overlapping community detection, Temporal analysis