Enhancing User Experience in User-Generated Content Websites by Exploiting Wikipedia

LIU CHEN

NATIONAL UNIVERSITY OF SINGAPORE
2013

Enhancing User Experience in User-Generated Content Websites by Exploiting Wikipedia

LIU CHEN

Bachelor of Engineering

Xi'an Jiaotong University

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2013

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Chen Liu

August, 2013

ACKNOWLEDGEMENT

I would like to thank my PhD thesis committee members, Anthony K.H. Tung, Sung Wing Kin, and Hsu Wynne for their valuable suggestions, comments and advice on my thesis. My first and foremost thank goes to my thesis supervisor Prof. Anthony K.H. Tung, who has introduced me to research. I still remember the first day I met Prof.Tung, when I expressed my willing to study as a PhD student in his office. I will always appreciate that he provides me such an opportunity. During the last half decade, I am deeply impressed by his insights and rigorous attitudes in research. All these are invaluable influences not only on my research but also my future life.

Prof. Ooi Beng Chin is another prominent figure in my life. He sets a high standard for our database research group, insists on the importance of hard working and advocates the value of building real systems. It is my great honor to be his research assistant for one and half a year. His speaks and behaviors empower my growth as a researcher and person.

The last five years in National University of Singapore have been an exciting and delightful journey in my life. It is my great pleasure to work and live with my friends, including Zhifeng Bao, Yu Cao, Ding Chen, Liang Chen, Yueguo Chen, Bingtian Dai, Wei Kang, Yuting Lin, Meiyu Lu, Feng Li, Peng Lu, Xuan Liu, Dhaval Patel, Zhan Su, Nan Wang, Tao Wang, Xiaoli Wang, Huayu Wu, Sai Wu, Wei Wu, Xiaoyan Yang, Shanshan Ying, Dongxiang Zhang, Jingbo Zhang, Meihui Zhang, Feng Zhao, Yuxin Zheng, and Jingbo Zhou. In addition, I would like to send my best regards to my friends who are not in NUS for those wonderful times we have together, including Da Li, Bing Liang, Chen Pang, Jilian Zhang.

Lastly but not least, I will always be indebted to my parents Jingliang Liu and Mingyan Qin. Their unconditional love has brought me into the world and developed me into a person with endless faith and power. Without their support, I can not go so far. Finally, my deepest love are always reserved for my wife, Li Zhang for accompanying me in the last eight years.

CONTENTS

SUMMARY

We have witnessed the incredible popularity of user-generated contents (UGC) over the last few years. The characteristics of UGC reflect that content production is no longer dominated by only a few experts or administrator. It becomes accessible and affordable to the general public through advanced techniques. Benefited from UGC, current Web is vastly enriched by articles, photos or videos created by ordinary users. For example, most contents of many leading websites, e.g., Facebook or Youtube are contributed by their users. Moreover, as stated in [7], UGC is the key to the success of many Web 2.0 services which encourages the publishing of one's own ideas and comments.

The advent of this new content production paradigm has brought several challenges:

1. UGC is very flexible and expressed in different formats, e.g., documents, photos or videos. Since they are represented in distinct space, browsing and retrieving of different kinds of UGC seems to be intractable.

2. Considering the vast information users may receive everyday from UGC websites, they probably face a serious "Information Overflow" problem. Therefore, to access the information more efficiently and effectively, an alternative browsing interface is required by users.

3. Since ordinary users are the creator of most UGC, their opinions, behaviors or interests are recorded and reflected from the contents to some extent. If accurate user model can be learned from UGC, a wide range of applications will be benefited, including personalized recommendations and online advertising.

In order to solve the first problem, we have proposed to represent UGC from different domains in the same Wikipedia space. Given the uniform representation, cross domain queries and search are supported. In this case, a variety of UGC is seamlessly integrated. Then user experience of browsing and retrieving of UGC is expected to be improved.

With respect to the second problem, we have designed a Wikipedia based method to organize and manage the information flow. In this method, contents on similar topics are grouped together so that users are able to easily identify newsfeeds that they are interested at. In addition, we further rank the grouped contents according to the user preference and distinguish them by explicit labels.

Finally, taking the user tag prediction as an example, we have investigated the usefulness of various sources of information in UGC websites. We offer a unified framework that takes into account the frequency, temporal, correlation and social information. Then based on the framework, we build a promising user tag prediction model.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Definition of User-Generated Content

UGC, also known as "User-Created Content" or "Consumer-Generated Media", can be broadly defined as anything amateur users produce. One formal definition of UGC by OECD [90] is as follows: "1) Content made publicly available over the Internet, 2) Which reflects a certain amount of creative effort and 3) Which is created outside of professional routines and practices". This definition reflects the main characteristics shared by very different content types published by Web users.

Entering the 21st century, the notion of User-Generated Content (UGC) is ubiquitous, especially thanks to a plethora of technologies. Since users are entitled with enormous rights, a variety of UGC format are developed, as listed in Table 1.1. Meanwhile, online services such as Facebook [1], Wikipedia [2], Twitter [3], Blogger [4], and YouTube [5] have established viable business models based on UGC. On these websites, users can publish their own diaries, post photos or videos, express opinions, discuss with other users and establish communities based on shared interests.

[1] www.facebook.com
[2] www.wikipedia.org
[3] www.twitter.com
[4] www.blogger.com
[5] www.youtube.com

Discussion boards	Blogs	Wikis	Social networking sites
Advertising	Public relations	Fanfiction	News sites
Trip planners	Memories	Mobile Photos, Videos	Customer review sites
Audio	Video games	maps and location systems	

Table 1.1: Different Types of UGC

To give reviewers a comprehensive and deep impression of UGC, we take Wikipedia as an example to introduce characteristics of UGC.

1.1.1 A Case Study: Wikipedia

Wikipedia is a perfect example of a UGC-driven website that shows an immense growth since its creation in 2001. As stated in the website, Wikipedia is a collaboratively edited, multilingual, free Internet encyclopaedia supported by the nonfprofit Wikimedia Foundation.

As a UGC website, one feature of Wikipedia is that almost all its contents, 30 million articles in total, over 4.3 million in the English Wikipedia, are written collaboratively by contributors around the world. In Wikipedia, anyone can write an article with sufficient expertise. Then the article can be edited by the other users with access to the site. The editions include evaluating the content, suggesting changes, or even making changes. Currently, it has about 100,000 active contributors. The second feature of Wikipedia is that the contents are free. This can be explained in two folds. 1) The contents are written by users voluntarily. Nobody will be paid for the knowledge he contributed. 2) Everyone can access the contents freely. There is no charge for obtaining information from the website. This feature can be found quite easily in many other UGC websites, such as Facebook, and YouTube.

So far, Wikipedia has shown significant influence on our everyday life. As of June 2013, Wikipedia is the seventh most popular website worldwide according to Alexa Internet, receiving more than 2.7 billion US pageviews every month, out of a global monthly total of over 12 billion pageviews. It has become the largest and most popular general reference work on the Internet. The growth of Wikipedia has been fuelled by its dominant position in Google search results; about 50% of search engine traffic to Wikipedia comes from Google. All these numbers demonstrate the great success of Wikipedia as well as the significant impacts of UGC sites.

1.2 Motivation for UGC Production

The phenomenal success of UGC has been well studied in the past [80]. We summarize and list several reasons in the following.

From the website perspective:

1. *Enrich User Experience.* The inclusion of UGC will increase the time spent on browsing in the website, as interesting and fresh opinions and comments will be more enjoyable and informative than just reading a sales pitch about a product or service. Therefore, it can increase the user loyalty to the website.

2. *Promote Business More Effectively.* UGC provides a promising way of interacting with potential customers on a personal level. Having UGC, the website is able to collect the user feedback more effectively and efficiently, thereby improving the services it provides. In addition, with user's participation, the website's online reach could be increased significantly.

3. *Increase Website Ranking.* UGC can keep fresh contents continually appearing on the website. This results positive effects on the search engine optimization of the website since current search engines emphasize more on the freshness of the contents on the site.

From the user perspective:

1. *Psychological Incentives.* In UGC sites, users are the principal involving participants. They have excellent flexibilities to express their opinions and participate in the websites. These actions allow the user to feel satisfactory as an active member of the community or keep relationships with others. In the meantime, other common social incentives are status, badges or levels within the site, something a user earns when they reach a certain level of participation. Another important psychological motivation comes from trust between users. UGC provides a trustworthy and personal source of information about an experience or venue.

2. *Explicit Incentives.* These incentives refer to tangible rewards. Examples include financial payments, an entry into a contest, a voucher, a coupon, or frequent traveller miles. For example, in Amazon Mechanical Turk, users expect to receive financial rewards after they fulfil a posted job. This financial incentive encourages user participation.

1.3 Research Problem: Enhancing User Experience in UGC Websites

Most previous efforts are spent on building platforms to attract more users. With the growth of UGC websites, now the problem turns to how to explore existing data to enhance user experience in these sites. In this thesis, we plan to give priorities to the following three aspects.

1.3.1 Cross Domain Search

Since UGC is expressed in a variety of format, the investigation of them is restricted on each single domain. We believe if we can integrate resources from different domains together, many useful applications can be built and user experience in these websites could be enriched. For example, while a user browses an image in Flickr, he possibly would like to read some related documents from other UGC websites, e.g., Blogger as a complementary explanation. The ideal scenario at that time could be the websites can automatically push information to the user. Then this can be assumed as a retrieval problem, where the Flickr image as a query, and documents from other sites as results . We show this in Figure 1.1. The query image is listed on the left while relevant documents are on the right.

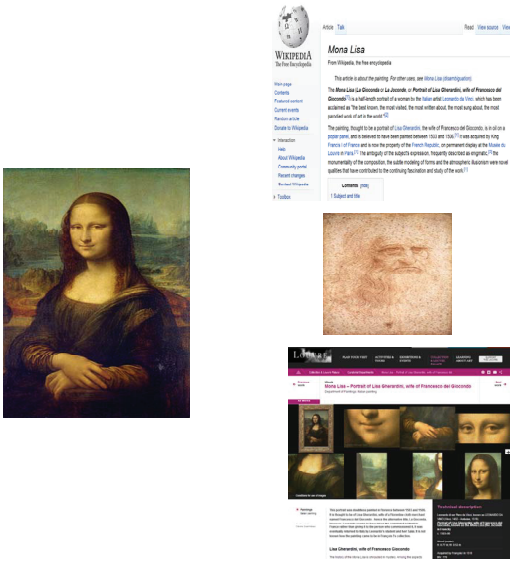


Figure 1.1: Cross Domain Scenario

Unfortunately, there are few attempts to integrate different UGC websites so far. The challenges of establishing such links are that the newsfeeds in different websites are not

in the same format. For example, Blogger typically hosts documents, Flickr for images and YouTube for videos. In this case, given an image from Flickr, it is a challenge to find its related documents in Blogger or videos in YouTube?

The problems of cross domain retrieval have been the subject of extensive research in areas such as information retrieval, computer vision, and multimedia [23, 81, 82, 66]. Generally, the existing solutions could be categorized into two groups. The first group is annotation based methods, in which all resources are assigned with several keywords. Then different kinds of resources are unified into a keyword space. However, this kind of method is only applicable in small data set and their feasibility in large web scale data is still in doubt. The second approach is uniform feature space based methods. It mainly performs the statistical analysis of features from different domains. However, this method requires large scale training data to learn the correlations between resources. Thus, the process will be time consuming and produce very high computation cost.

In this thesis, we address the above problem similar as the first approach. The observation is that different kinds of resources are usually annotated with tags. However, compared to the expert-edited data set used in previous work, these tags are much more noisy and shorter. The main contribution of our work is that we propose to utilize Wikipedia to clean the user contributed tags and add map them to the Wikipedia concept space. Finally, users can submit various format of resources as queries, (e.g., keyword queries, image queries or audio queries) and the system will return the corresponding results from different websites. By implementing this, a uniform Cross Domain Search (CDS) framework is set up which enables users to seamlessly explore and utilize the massive amount of resources distributed across different UGC services.

1.3.2 A Personalized Knowledge View for Twitter

On UGC websites, the flood of information stream for a user becomes highly intensive and noisy, users are faced with a “needle in a haystack” challenge when they wish to read the news that is interesting to them. Along with this problem, the users’ reading experience will be hurt and their time will be wasted.

Many approaches have been proposed to handle this problem. The common methods consider how to arrange the positions of newsfeeds. Intuitively, users prefer feeds that they are interested at to be ranked higher so that they can find it more easily. However, most current websites simply order the newsfeeds based on their freshness, the newly

Figure 1.2: Facebook Interface

published resources are ranked higher in the interface. People try to improve the orders through multiple ways [16, 8]. One typical newsfeed ranking algorithm among them is developed by Facebook, called "EdgeRank" [48]. An Edge is basically everything that "happens" in Facebook. Examples of Edges would be status updates, comments, likes, and shares. There are many more Edges than the examples above; any action that happens within Facebook is an Edge. EdgeRank ranks Edges in the newsfeeds. The evaluation of an edge is made up of affinity, weight, and time decay. Affinity is a one-way relationship between a user and an Edge. It could be understood as how close of a "relationship" a brand and a fan may have. Affinity is built by repeat interactions with a Brand's Edges. Actions such as commenting, liking, sharing, clicking, and even messaging can influence a user's affinity. EdgeRank looks at all of the Edges that are connected to the user, then ranks each Edge based on importance to the user. Weight is a value system created by Facebook to increase/decrease the value of certain actions within Facebook. Commenting is more involved and deemed more valuable than a "like". In this system, all Edges are assigned a value chosen by Facebook. Time Decay refers to how long the Edge has been alive; the older it is, the less valuable it is. Summarizing all factors above, objects with the highest EdgeRank will typically go to the top of the newsfeeds. We present an exemplar interface of Facebook in Figure 1.2.

In this thesis, we propose a different solution for this problem. Besides the order, we present users a new display of the newsfeeds. We expect to visualize the stream in a knowledge view, which is shown in Figure 1.3. In the knowledge view, we present users an overview of their tweet stream as shown in the node labels. Thus, users can catch what is happening around them. Furthermore, each hierarchical tree represents tweets

with similar topics. Following the hierarchy, users are able to find topics that they are interested quickly and read tweets topic by topic. By clicking on the node of the tree, related tweets of the node label will be popped up.

We continue to use Wikipedia to develop the techniques. Moreover, besides its concepts, we additionally build a knowledge graph based on Wikipedia's hierarchical relations. Then the connections between tweets are learned from the graph. In addition, we propose several evaluation metrics to rank the generated subtrees and achieve the personalized arrangement by devising a kernel function.

Figure 1.3: An Alternative Interface for Twitter

1.3.3 User Tag Modeling and Prediction

The first two works investigate UGC from an explicit view, including supporting cross domain search and building an alternative interface. However, a more intrinsic problem we need to address is how we can understand users better. With the growth of UGC, this issue plays an essential role in providing personalized services for users.

In this thesis, we take a typical user interest prediction problem, user tag prediction, as an example to illustrate our trial in handling the above issue. Tag prediction receives quite a lot of attention recently since the tag data is easy to obtain and the results can be evaluated in a relatively straightforward way. The traditional definition of tag prediction is usually about *resource tag prediction*, which is to predict relevant tags to a particular item. Researchers have proposed collaborative filtering methods [2] or graph based methods [67] to address the problem. However, in resource tag prediction, tags are deemed as

annotations of resources. This neglects tags also existing as a denotation of user interests. Therefore we propose a new definition of the tag prediction, which is *user tag prediction*. The prediction of user tags is based on the user instead of the resource. Compared to the resource tag prediction, we focus more on understanding how interests of users evolve and what are their behavior patterns. The applications of resource and user tag prediction are different as well. With successful user tag prediction, different types of contents can be pushed to the right people, thereby satisfy their personal demands.

We present a unified framework for predicting user tags in social network. Our proposed framework learns a prediction model for each user using four types of information namely, (i) *Frequency:* frequency of tags in the user's profile if there is any, (ii) *Temporal:* temporal usage or patterns of a tag, (iii) *Correlation:* the correlations of a tag with its surrounding tags, and (iv) *Social Influence:* the social influence imposed from the user's social network. We have analytically studied the prediction power of each source. Finally, we extract a number of features according to the analysis and model the prediction as a classification problem.

1.4 Contributions of this Thesis

In this thesis, we primarily address the problems of enhancing user experience in current UGC websites. First, we establish a cross domain search framework to incorporate resources from different UGC websites. Such a strategy bridges gaps between UGC sites and fully utilizes the data across websites. Consequently, users are able to retrieve information more conveniently. Second, we design an alternative interface for users to view the newsfeeds in UGC sites. In the new interface, we categorize the information flow into several groups based on its topics. Each group is organized in a hierarchical style with each level clearly labelled. Additionally, these grouped contents are ordered according to user preferences. In general, in designing solutions to the above two problems, we integrate the semantics in Wikipedia into the representation of UGC in order to obtain a better understanding of the data. Lastly, we utilize the personal profile collected in UGC websites to perform user interest modelling and prediction. This is consistent with the trend in current UGC systems, which is to provide better personalized service. As a start, we use user tag prediction as the example and investigate the problem from four perspectives: frequency, temporal, correlation and social influence. We perform in-depth analysis on information regarding to those perspectives and extract related features based on the analysis.

1.5 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2 reviews the research related to this thesis. The surveyed topics include cross domain search, organizations of resources, user interest modeling and Wikipedia applications.
- Chapter 3 presents the detailed description of establishing cross domain search framework. The highlight is various types of resources are uniformly represented in the same space.
- Chapter 4 presents our strategy to organize information flow for users in UGC websites. A hierarchical interface will be built to present them the newsfeeds.
- Chapter 5 presents our framework for user tag prediction. Numerous features will be constructed according to our analysis on the user profile.
- Chapter 6 concludes this thesis and lists several future research directions on the topic of improving user experience in UGC websites.

CHAPTER 2

RELATED WORK

Interests in analysis of user-generated contents have grown massively in recent years. Researchers are attracted by the large scale and rich personal data contained in UGC websites. The studies on the data are applicable to many areas, which are almost intractable to name them all. In this chapter, we will give a literature review on related research and systems. First, since a wide range of contents is included in UGC sites, we will give an overview of studies and applications on cross domain search in Section 2.1. Second, we will introduce more details on how to organize resources in Section 2.2. The general idea of these works is to improve the efficiency and effectiveness of users' reading. In particular, we will use Twitter as an example to illustrate related research. Third, related works on user interest modeling and prediction will be discussed in Section 2.3. Finally, we will give a brief introduction of related works about Wikipedia since it is a critical knowledge base through the whole thesis.

2.1 Cross Domain Search

Nowadays, resource retrieval in single domain is well studied. For example, in the text domain, a great success has been achieved by PageRank [11] type methods in Google.

In image domain, based on content based image retrieval (CBIR) [23] techniques, many systems have been built. In a project developed in [95], users can search for aviation photos by submitting a photo. However, since UGC covers a wide range of media contents, simply resource retrieval from single domain if far from enough. Since the primary components of UGC are text or images, we will use these two as examples to illustrate how to approach cross domain search by existing methodologies.

One of the challenges in cross domain search is that there is no corresponding links between resources in different formats. Although various features are developed for documents or images [23], resources in different domains are still expressed in different features, e.g., documents in text features and images in visual features. To support cross domain search, the connections between the two space need to be identified, or the "semantic gap" [83] need to be filled.

A general idea of solving this problem is to learn the correlations between the two space, or map them to the same space. The first kind of methodology is machine learning oriented methods. These methods aim to learn mapping functions or transform data into a common space, then traditional similarity functions such as cosine similarity can be applied. For example, [12, 49, 108] project the original data into the hamming space using different learning methods, such as AdaBoost [12], probabilistic graphical model [108] or project matrix [49]. However, the main drawback of these methods is that the mapped space does not contain semantic information. [72] expects to make an improvement by incorporating the semantic information and using canonical correlation analysis (CCA) to project the image and text features into a common d-dimensional subspace. [46] builds a Markov random field of topic models, which connects the documents based on their similarities. As a consequence, the topics learned with the model are shared across connected documents, thus encoding the relations between different modalities. These machine learning oriented methods are limited to domain specific applications, such as shape retrieval [49], due to the requirement of training data set and high computation cost. As a result, they are not suitable for UGC, which is noisy and has a vast volume.

The second kind of methods is based on automatic image annotation technologies. They try to map images to text space so that images will have the same features as documents. The prerequisite of automatic image annotation is that it asks for a training data set which includes images which are already associated with a set of keywords. Then for an image without any annotations, by applying CBIR, relevant keywords will be generated and attached to the image. In this way, images and text are represented in the same space. The existing annotation approaches can be classed into three categories: classification based

[50], probabilistic model based [37, 45] and search based [77]. Based on this kind of methods, many applications have been built. In [28] and [102], they propose techniques of recognizing locations and searching the Web documents by certain images.

At the early stage of automatic image annotation, the training data set is very small since all the keywords are manually generated by experts. This limitation restricts the practical use of this kind of methods. However, with the popularity of Web 2.0 technologies, more large and real data set is available to work upon since more users on the Web contribute a large volume of such kind of contents, e.g., Flickr images. These big data has prompted studies on verifying or developing new approaches to handle this problem. In [60], they have utilized a similar method as before but conduct the experiments on a relative larger data set. However, simply applying previous methods is problematic. The reason is that the associations between images and keywords are much more reliable in expert editing training dataset compared to the common user contributed contents. Since in the later case, the annotations are added by users arbitrarily, many of them are not related to the content of resources. Therefore, directly working on the user annotations will introduce many incorrect associations. To handle this problem, many researchers propose to utilize ontology, e.g., Open Directory Project (ODP), DBPedia [6] and Yago [38] to add more semantics to the annotations. For example, [93] presents the data in concept vectors, which are derived from ODP. [56] implements similar work by using Wikipedia. Except ontology, there also exist works which try to utilize related news articles [1] or web documents [78] to fulfill the same task.

In this thesis, we develop a Wikipedia based method to clean the annotation in advance. Then, we map the annotations to the Wikipedia concept space instead of the simple tag space in order to obtain more accurate descriptions.

2.2 Resource Organization

To enhance user experience in UGC websites, besides supporting cross domain search, how to improve users' reading efficiency is another problem. As the information stream floods in from different UGC websites, users are faced with a "needle in a haystack" challenge when they wish to read an interesting feeds. The solutions to this problem are quite critical to current services as they have to make sure users will not feel frustrated when look for useful information. Next, we list works from three areas which are assumed to be possible solutions.

2.2.1 Resource Re-Ranking

The first type of methods is resource re-ranking, which tries to order the resources according to different criteria. The intuition is to rank the resources which may be interesting to users in higher positions. As the service provider, Facebook uses the proprietary EdgeRank[48] algorithm that decides which stories appear in each user's newsfeeds. The algorithm hides boring stories, so if your story does not score well, no one will see it. In particular, this algorithm favors recent, long conversations related to close friends. Comparatively, Twitter adopts simple rules to filter correspondences between users aggressively. These services also provide options to users that only display information from particular lists or tags. From the research perspective, most related works aim to provide a personal ranking mechanism so that interesting tweets will be displayed first. For example, [34] re-ranks the stream based on three dimensions: people, terms, and places. Their results indicate that user's stream data could be an effective source for the personalization task. [17] studies the URL recommendation on Twitter with the purpose of guiding users' attention better. They evaluate several algorithms through a controlled field study and conclude that both topic relevance and social voting are useful for recommendations. In their following work [16], they utilize the thread length, topic and tie-strength to evaluate the feed importance. [27] extracts a set of features, such as the content relevancy and account authority, from Twitter feeds, then they employ a machine-learned ranking strategy to determine the ranking of feeds. In [39], they address this problem as an intersection of learning to rank, collaborative filtering, and click-through modeling. They propose a probabilistic latent factor model with regressions on features extracted from contents.

2.2.2 Topic Detection

Topic is another key to organize resources in a meaningful way. With the topic denotation, users are able to understand the resources quickly and find other resources with similar topics. In order to discover topics from tweets, [65] characterizes contents on Twitter via a manual coding of tweets into various categories, e.g., from "Information Sharing" to "Self-Promotion". The popular topic modeling methods, such as Latent Dirichlet Allocation(LDA), are introduced as well to describe the resources in the topic level. In LDA, each document is represented as a probability distribution over some topics, while each topic is represented as a probability distribution over a number of words. [40] proposes several schemes to train a standard topic model on tweets and further compares their

quality and effectiveness. While in [71] characterizes users and tweets using labeled L-DA, which is a semi-supervised learning model, to map tweets into multiple dimensions, including substance, style, status, and social characteristics of posts. Furthermore, [8] utilizes search engines to enrich tweets with more relevant keywords in order to capture the topics of the tweet stream. Recently, [62] proposes the problem of linking each tweet with related Wikipedia articles. This is achieved by extracting features from tweets and then using machine learning methods to determine the correct concepts from tweets. Their results yield that the bag of concepts representation assists in understanding the text.

2.2.3 Visualization Tools

The third kind of approaches to above problem is to design new meaningful interfaces for users to explore resources. Currently, resources are usually ordered in a single list. More visualization tools need to be developed to view the resources from different perspectives. Next, we list several interesting systems. [26] designs an evolving, interactive and multi-faceted visualization tool to browse the events discussed in Twitter. Similarly, Vox Civitas [25] is a visual analytic tool to help journalists to extract the most valuable news from large-scale social media contents. Recently, TwitInfo [61] is proposed to generate an event summary dashboard, which identifies the event peak and provides a focus+context visualization of long running events. All these works manipulate the whole tweet corpus, aiming at the general population. There are many other works focusing on each individual user. For example, Eddi [8] groups each user's incoming stream based on their topics. Users are allowed to browse the tweets firstly on the topics they are most interested at. In addition, there also exist some works [52, 58] which try to extract keywords from the contents. Then tag visualization techniques are applied to the keywords to help users to obtain a clearer understanding.

In summary, various approaches are developed to cope with the problem. In this thesis, our method is a combination of previous efforts. We intend to provide users a knowledge view of the resources, thus make it a topic driven method since [16] pointed that most users are more sensitive of the content topics. We extract topics of resources based on Wikipedia. Furthermore, we build a new visualization interface for resources. In the interface, resources are organized by several hierarchical trees with each node identified by its label. Lastly, the trees are ordered according to users' preferences.

2.3 User Interest Modeling & Prediction

Modeling user interests is the common practice for many applications, such as search engines, recommendations, online advertisements and etc. With the growth of UGC websites, more personalized service are required. Motivated by these needs, researchers have considered various methodologies to capture the user's interests. Early works on modeling user interests, such as [18] and [59], have utilized the interests which are explicitly specified in users' profiles. Recent works model the interest by analyzing their behaviors, which are observed from various sources. They assume that user interests are implicitly embodied in their activities. A significant portion of sources is from the search context, such as the search history, browsing activities and etc [84, 98, 69]. In these works, different kinds of user profiles and models are built to improve the personalized search. Furthermore, topic information is also utilized to build the interest model. The Open Directory Project and Wikipedia are the two typical sources to obtain topic semantics. Latent Dirichlet Allocation is the most widely used technique to model user interests [99, 3].

The rich personal data in UGC websites has provided new sources to perform the task. As stated in [51], there are two kinds of approaches to discover user interests in social networks. The first kind is user-centric, which focuses on detecting interests based on social connections among users. Various approaches have been developed to infer user interests and make predictions [79, 97, 4]. They mainly investigate of the social connections or the influence to each other. One of the basic ideas of these works is collaborative filtering, which suggests that a user will be interested at his neighbors' recommendations. The second type is content-centric, which detects interests based on contents contributed by users to a social community. For example, [74] and [105] take users' tagging behaviors into account to improve the personalized recommendation. Similarly, in [51], they utilize the association rule mining to discover interests from user generated tags. Besides tag behaviors, tweets are utilized as well to implement recommendation [68]. Another user interest modeling work for Twitter is developed in [1]. Their approach depends on matching tweets to news articles, then the tweets are added more semantics by the news article's content. Finally, these tweets are used for user interest modeling. There also exists work which tries to combine both user connections and contents to handle the task. In [93], user generated contents and neighbors are uniformly represented as concept vectors. Then user interests are learned from heterogeneous sources and concept associations.

In above works, user interests are usually assumed to be constant over time. Apparently, it is not a well-accepted premise. Several works have been done to demonstrate that

temporal effect plays an essential role in determining user interests. For example, in [104], the performance of tag prediction is improved when temporal dynamics of user interests are incorporated. Similar findings are suggested in [3], as well. However, in these works, temporal dynamics are simply assumed to decay exponentially.

2.3.1 Tag Prediction

As a typical interest prediction problem, tag perdition is well-studied in the literature. There are three approaches for personalized tag predictions in social tagging systems: (1) Content-based methods [54, 105], which model users' interests from the contents of resources and user profiles. One advantage of these methods is that they can predict tags for new users and new resources. One state of the art content-based method is proposed in [54]. They utilize the user history and resource contents to build profiles for both users and tags. Then a set of tags, which are related to both the resource and the user are predicted. (2) Graph-based methods. We further divide this approach in two sub-directions. The first one is the user/resource based collaborative filtering [67, 53, 107]. They build a matrix between users and tags or resources and tags and learns the relation of tags to users from the matrix. The second one is the random walk methods [29, 103]. In this method, they first build a graph among users, tags and resources. Then either supervised or unsupervised random walk algorithms are run on the graph. The differences of them are mainly about the edge type, edge weights and etc. (3) Tensor decomposition methods [74, 75, 86]. The relations of users, resources and tags are modeled as a cube with many unknown entries. Then the unknown entries could be predicted by low-rank approximations after performing tensor decomposition. However, this approach is usually quite expensive since the smoothed user-resource-tag tensor which is obtained by high order SVD, is usually not sparse.

Different with previous works, in this thesis, we primarily investigate the user tag prediction problem, in which we would like to predict which tags the user will use in the future. We apply a machine learning method to solve the problem. Many applications could be benefited from this study, such as social network advertising and content recommendation.

2.4 Wikipedia & Its Applications

One of the core components of this thesis is the utilization of Wikipedia. Wikipedia is a free, collaboratively edited and multilingual Internet encyclopedia. It seeks to create a summary of all human knowledge in the form of an online encyclopedia, with each topic of knowledge in one article. The crowdsourcing writing manner makes it become the largest and most popular general reference work on the Web. Wikipedia has attracted a lot of attentions from researchers because of its high quality data, such as its articles, info box, relations between articles, multilingual properties or its community structures [100, 88, 10, 55]. In this thesis, semantics of Wikipedia are employed in many places and have different roles. Next, we will separately list the related works.

1. *Data Linkage:* Due to its broad covering scope of entities, it becomes a natural hub for connecting data sets. For example, DBpedia [6], which is derived from Wikipedia is interlinked on the RDF level with various other open data sets on the Web. This enables applications to enrich DBpedia with data from those data sets. As of January 2011, there are more than 6.5 million interlinks between DBpedia and external data sets. Besides above works, several applications [19, 92] have recently become publicly available that allow connecting ordinary Web data to Wikipedia, in both automated or semi-automated fashion. A project [92] prompted by BBC tries to provide background information, which is derived from Wikipedia on identified main actors in the BBC news. Alternatively, in [19], a service of linking Flickr images to Wikipedia is described. In this thesis, resources from different domains are connected to Wikipedia concepts to support cross domain search. Compared to above works, our method fits for a more general framework.

2. *Name Entity Disambiguation :* The redirected pages, disambiguation pages and abundance of links embedded in Wikipedia provide rich sources in disambiguating name entities. The studies in [64, 63, 22] exploit a similar framework. They only differentiate each other in the implementation of each step. This framework consists of three steps, 1) collect all the possible entities for text. 2) identify possible name entities. 3) figure out the most appropriate entity name in the corresponding context. While mapping tags to Wikipedia concepts, we apply similar techniques. With the name entity disambiguation, the meaning of tags could be captured more precisely.

3. *Text Clustering and Classification:* In these two tasks, Wikipedia is more often used as a semantic enrichment tool so that similarities between text are expected to be computed more precisely. For the clustering task, [41] extends the Wikipedia

concept vector for each document with synonyms and associative concepts. This information is collected from the redirect links and hyperlinks in Wikipedia. Then they improve the document similarity measure by including those related concepts. Compared to [41], [43] implements one step more by including category information in Wikipedia. Another related issue is the label generation for clusters. For produced clusters, [13] extracts and evaluates concepts from Wikipedia as the labels. The shortfalls of their methods are that the clustering and label generation are totally separated. Our intuition is that a good clustering method could help us find better labels for clusters and vice versa. Based on this intuition, we intend to develop a joint method which can integrate the two processes together.

With respect to the classification task, [30] performs text feature generation using a multi-resolution approach: features are generated for each document at the level of individual words, sentences, paragraphs, and finally the entire document. This feature generation procedure acts similarly as a retrieval process: it receives a text fragment (such as words, a sentence, a paragraph, or the whole document) as input, and then maps it to the most relevant Wikipedia articles. [96] builds a kernel function to compute the similarity between the Wikipedia enriched documents.

4. *Wikipedia Concept Relatedness:* Since we primarily describe resources by Wikipedia concepts, understanding relatedness of concepts is a good way to figure out the correlation between resources. The state of the art concept relatedness computation is described in [31]. For each term, it first retrieves all the articles that contain the term. For each article, a $TFIDF$ score is computed between the term and that article. All the computed scores are used to construct a high dimensional vector for the term. At last, the relatedness between two terms are directly computed by the cosine similarity between article vectors. Although this method is robust, it is time consuming. To improve the efficiency, [64] only utilizes the Wikipedia link structure and computes the similarity by applying the Normalized Google Distance.

CHAPTER 3

CROSS DOMAIN SEARCH BY EXPLOITING WIKIPEDIA

The abundance of user-generated contents in various media formats calls for better integration to utilize resources fully. This naturally leads to a new wish of resource search in different domains. For example, given a document, if we can automatically assign Flickr images to it, it will be very attractive for users to read.

The first requirement to make the integration become true is to find a common link between resources. Unfortunately, the integration of UGC systems is hard to implement as most current systems do not support cross reference to related resources of each other. The difficulty of establishing such cross-system links is how to define a proper mapping function for the resources from different domains. For instance, given an image from Flickr, we need a function to measure its similarities to the documents in Delicious or videos in YouTube.

The problem of building connections between images and text has been well-studied [23, 72, 60, 70] in the past. However, these methods always incur high computation overhead and involve complex learning algorithms. Most of the existing schemes are not

flexible and cannot be extended to support an arbitrary domain. Namely, they are limited to a single format. If a new UGC website emerges, the scheme needs to be fully redesigned to support it, which is not scalable for the real system. Compared to extracting internal features from resources as above, we observe that tags are widely used to annotate resources across UGC websites. Therefore, acting as a straightforward medium, these tags connect resources from different domains together. However, tagging is by nature an ad hoc activity. They often contain noises and are affected by the subjective inclination of the user. Consequently, linking resources simply by tags will not be reliable. In this chapter, we propose to utilize Wikipedia, the largest online encyclopedia, to build a concept layer to connect resources. Within the concept representation of resources, users are able to explore and exploit seamlessly the vast amount of resources which are distributed across UGC websites. In this framework, the user can input queries from any specific domain (e.g., keyword queries, image queries or audio queries) and the system will return the corresponding results from related domains.

We introduce the preliminary knowledge and the proposed system architecture in the next two sections. Then, we explain how to select the useful tags and build the uniform concept vector in Section 3.2. In Section 3.3, we present our retrieval method for cross domain resources. Finally, we report our experimental study and conclude this chapter in Section 3.5.

3.1 Problem Statement

Various types of UGC websites are established to serve different groups of users. For example, Flickr is the first choice for photographers to share their products while YouTube is the playground for the video lovers. To enhance the users' experience, most of these websites provide an in-site search service, which exploits the state of the art information retrieval techniques to improve the search results. However, to our knowledge, none of current sites provides an integrated search that allows the user to search all UGC websites by various queries in different modalities via a single search portal as it is challenging to provide a universal metric to link and rank different types of resources.

Formally, each website can be considered as a domain, $\mathcal{D}$. Given two resources r_i and r_j in the same domain, we have a function $f(r_i, r_j)$ to evaluate how similar the two resources are, but if r_i and r_j come from different domains, no such similarity function exists. The idea of this chapter is to exploit the semantics of Wikipedia to connect different domains

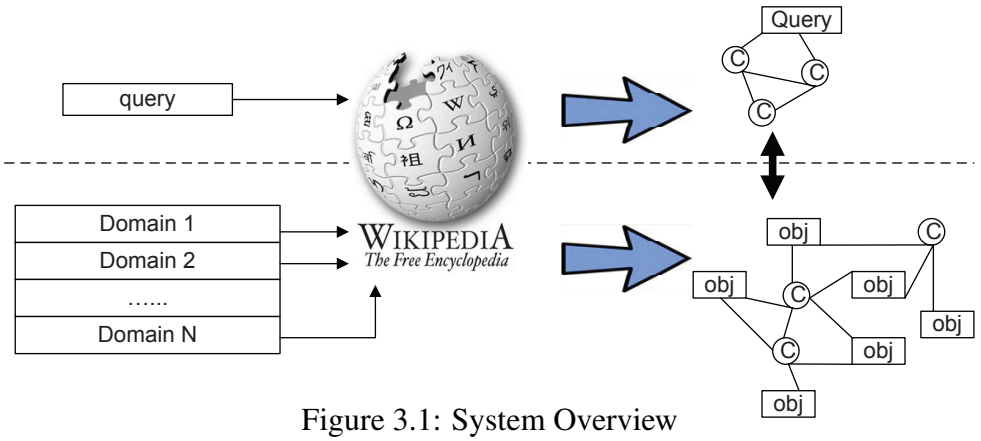


Figure 3.1: System Overview

Very large database

From Wikipedia, the free encyclopedia
(Redirected from Vldb)

A **very large database**, or **VLDB**, is a database that contains an extremely high number of tuples (database rows), or occupies an extremely large physical filesystem storage space. The most common definition of VLDB is a database that occupies more than 1 terabyte or contains several billion rows, although naturally this definition changes over time.

Figure 3.2: Wikipedia Snippet

so as to support cross domain search. Wikipedia is the largest online encyclopaedia and to this date is still growing with newsworthy events. New topics are often added within a few days. It contains more than 2 million entries, referred as Wikipedia concepts throughout this thesis. Most of them are representative name entities and keywords of different domains. The size of Wikipedia guarantees the coverage of various resources in different domains. In addition, the rich semantics and collaboratively defined concepts could help us to relieve the disambiguation and noisy problems in the description of resources.

Figure 3.1 shows the overview of the system. For each Web 2.0 system, we develop a crawler based on the provided API. All crawled resources are maintained in our data repository. We organize the resources by their domains and a similarity function is dened for each domain. In the ofine processing, resources from different domains are mapped to the Wikipedia concepts by their tags and represented by the uniform resource vectors. In this way, resources from different domains are linked via the concepts. Given a query, which may be issued to an arbitrary domain, the query processor translates it into the uniform concept vector via the same approach. Then, query processing is transformed into matching resources of similar concepts, which can search all domains seamlessly.

3.1.1 Wikipedia Concept

In this work, each Wikipedia article is considered as the description of a concept. The title of the article is used as the name of the concept. For example, Figure 3.2 shows the article about concept "VLDB". In this way, the Wikipedia dataset can be considered as a collection of concepts, $\mathcal{C}$. We use $D(c)$ to denote the article of concept $c \in \mathcal{C}$. To catch the semantics between concepts, we define three relationships for them.

The first relationship is defined based on the article structures of Wikipedia.

Definition 3.1. Link between Tag and Concept
In a Wikipedia article $D(c_i)$, if tag t is used to refer to another concept c_j, t is linked to c_j and we use $t \rightsquigarrow c_j$ to denote the relationship.

In Figure 3.2, tag $tuples$ and $filesystem$ are used to refer to the concept "Tuple" and "File System", respectively. Therefore, we have $tuples \rightsquigarrow Tuple$ and $filesystem \rightsquigarrow File\ System$. If $t \rightsquigarrow c_i$, when clicking t, Wikipedia will jump to the article $D(c_i)$. This behavior is similar to the hyperlinks between web pages. As Wikipedia articles are created by the Internet authors, who may use different tags to describe the same concept, multiple tags are probably linked to one concept. Then, we can have $t_x \rightsquigarrow c_i$ and $t_y \rightsquigarrow c_i$. To measure how closely a tag is related to a concept, we define $w(t_x \rightsquigarrow c_i)$ as how many times tag t_x is linked to c_i in Wikipedia.

The second relationship is used to track the correlations of concepts. The intuition is that if two concepts appear in the same article, they may be correlated with a high probability.

Definition 3.2. Correlation of Concepts
Concept c_i is said to be correlated with concept c_j, if

- $\exists t \in D(c_i) \rightarrow t \rightsquigarrow c_j$
- $\exists t \in D(c_j) \rightarrow t \rightsquigarrow c_i$
- *there is a document $D(c_0)$, satisfying $\exists t_1 \in D(c_0) \exists t_2 \in D(c_0)(t_1 \rightsquigarrow c_i \wedge t_2 \rightsquigarrow c_j)$*

Table 3.1: Wiki-DB Interface

Function Name	Description
$w(t_i \rightsquigarrow c_j)$	Return the link score between tag t_i and concept c_j
$P(c_j\|c_i)$	Return the correlation of c_j to c_i
$d_c(c_i, c_j)$	Return the semantic distance between c_i and c_j

Based on Figure 3.2, concept "VLDB", "Tuple", "File System" and "Terabyte" are correlated to each other. In particular, given two concepts, c_0 and c_1, we use

$$P(c_1|c_0) = \frac{\sum_{c_i \in \mathcal{C}} \theta(O(c_i), c_0) f(O(c_i), c_1)}{\sum_{c_i \in \mathcal{C}} \theta(O(c_i), c_0)}$$

to compute the correlation of c_1 to c_0. $\theta(O(c_i), c_j)$ returns 1 if there is a tag in $D(c_i)$ linking to c_j. Otherwise, $\theta(O(c_i), c_j)$ is set to 0.

The last relationship is derived from the hierarchy of Wikipedia. In Wikipedia, articles are organized as a tree according to the Wikipedia hierarchy. .[1] We use the tree distance to represent the semantic distance of concepts. To simplify the presentation, we use $L(c_i)$ to denote the level of the concept. Specifically, the level of root category "Articles" is 0. The semantic distance of concept c_i and c_j is defined as:

Definition 3.3. Semantic Distance
Given two concepts c_i and c_j, let c_0 be the lowest common ancestor of c_i and c_j. The semantic distance of c_i and c_j is computed as:

$$d_c(c_i, c_j) = L(c_i) + L(c_j) - 2L(c_0)$$

In this chapter, the above relationships are combined and used to rank the similarities between concepts and tags. However, it is costly to evaluate the similarities of concepts on the fly. Therefore, in the pre-processing, we scan and compute all the tag links, concept correlations and semantic distances. The pre-processing results are maintained in our MySQL database. We wrap the database searches to a set of simple interfaces as shown in Table I.

[1] Strictly speaking as some articles are classified into multiple categories, the concept graph is not a tree. In most cases, it can be processed as a tree.

3.2 Cross-Domain Concept Links

We develop customized crawlers for each website. The crawled resource (including images, videos and web pages) is abstracted as $(\mathcal{T}, \mathcal{V})$, where $\mathcal{T}$ denotes the tags assigned to the resource and $\mathcal{V}$ is the binary value of the resource. Intuitively, if two resources are associated with similar tags, they possibly describe the similar topics, regardless of their domains. However, as tags are normally input by the humans, which may contain noisy terms or even typos, they probably introduce misleading correspondences. For instance, the tag can have the homonym (a single tag with different meanings) and synonym (multiple tags for a single concept) problem. A resource tagged by "orange" may refer to a kind of fruit, or it just denotes the colour of an object. A picture of Apple's operating system possibly is tagged by "Leopard", which may be confused with the animal. To ease this problem, we propose to describe the resources in Wikipedia space instead of the tag space. Given the semantics in Wikipedia, we expect the description could be more accurate.

Given a concept space $\mathcal{C}$ in Wikipedia, our idea is to create a mapping function between the resources and concepts. The result is the uniform concept vector $v_i = (w_1, w_2, ..., w_n)$ for each resource r_i. w_j denotes the similarity between r_i and concept $c_j \in \mathcal{C}$ (namely, the weight of c_j). The mapping is constructed via the tag sets, which can be formally described as:

Definition 3.4. Mapping Function
The mapping function f is defined as $f = \mathcal{T} \times \mathcal{C} :\Rightarrow w_1 \times w_2 \times ... \times w_n$, where $n = |\mathcal{C}|$ and $0 \leq w_i \leq 1$.

Note that the uniform concept vectors are built upon tags. It is unreasonable to assume that all the tags are relevant to the resource content. People perhaps apply an excessive number of unrelated tags to resources such that the supposed unrelated resources will be connected by the spam tags. Therefore, in this section, we first discuss how we process the tag set $\mathcal{T}$ for each resource and then we introduce our tag-based mapping function.

3.2.1 Tag Selection

Several studies have been done on mapping the tags to Wikipedia concepts. The first category is quite a straightforward method. It leverages the power of existing search

engines, e.g., Google[2] to identify related concepts [78]. The second category utilizes Wikipedia to generate features of text fragments, e.g., explicit semantic analysis (ESA) [31, 42] and then look for the corresponding concepts. However, we argue that as people can tag the resources with arbitrary phrases, it is necessary to filter and remove the spam tags. As an example, in our collected Flickr data set, each image is tagged by about 10 unique tags. Many of them are ambiguous or unrelated to the central topic of the images. If all those tags are applied to search Wikipedia concepts, we will end up with too many unrelated concepts.

The tag selection affects the efficiency and accuracy of the mapping process. Based on the observation that correlated tags are normally used for one resource, we propose a cluster-based approach. Given a tag set $\mathcal{T}$, our idea is to group the tags into a few subsets: S_1, S_2,...,S_k. The subsets satisfy

$$\bigcup S_i = \mathcal{T} \& S_i \cap S_j = \emptyset$$

KL-divergence is used to measure the quality of clustering. In information retrieval, they have a theory that if the keywords are about similar topics, the query comprised of those keywords will get a better result. In the statistical point view, the returned pages will represent a significantly different keyword distribution with the whole collection [21]. We adopt the similar idea here. We expect the produced clusters could have a significantly different tag distribution in their associated Wikipedia articles compared to the whole article set. Let $\mathcal{W}$ and $\mathcal{C}$ denote the whole tag set and concept set of Wikipedia respectively. For two subsets, S_i and S_j, we have

$$KL(S_i, S_j) = \sum_{t_x \in \mathcal{T}} P(t_x|S_i \cup S_j) \log \frac{P(t_x|S_i \cup S_j)}{P(t_x|\mathcal{W})} \tag{3.1}$$

$P(t_x|\mathcal{W})$ is computed as the probability that tag t_x appears in all articles of Wikipedia

$$P(t_x|\mathcal{W}) = \frac{tf(t_x)}{\sum_{t_y \in \mathcal{W}} tf(t_y)}$$

while $P(t_x|S_i \cup S_j)$ is estimated using the following way.

We replace $S_i \cup S_j$ with their involved articles in the Wikipedia. In particular, for a tag $t_x \in S_i \cup S_j$, let $D(t_x)$ be the articles[3] that related to t_x, that is

[2] http://www.google.com

[3] $D(t_x)$ can be also considered as a set of concepts for the corresponding articles.

$D(t_x) = \bigcup D(c_i)\ where\ w(t_x \rightsquigarrow c_i) > 0.$

The whole involved articles are

$$D(S_i \cup S_j) = \bigcup_{t_x \in S_i \cup S_j} D(t_x)$$

Then, $P(t_x|S_i \cup S_j)$ is computed as:

$$P(t_x|S_i \cup S_j) = \sum_{t_y \in S_i \cup S_j} P(t_x|t_y)P(t_y|S_i \cup S_j)$$
$$= \sum_{t_y \in S_i \cup S_j} P(t_y|S_i \cup S_j) \sum_{c_z \in D(S_i \cup S_j)} P(t_x|c_z)P(c_z|t_y)$$

where c_z is a concept in the article set $D(S_i \cup S_j)$. To compute $P(t_x|c_z)$, we define $tf(t_x, c_z)$ as the term frequency of t_x with regard to the article of c_z.

$$P(t_x|c_z) = \frac{tf(t_x, c_z)}{\sum_{t' \in \mathcal{W}} tf(t', c_z)} \tag{3.2}$$

Recall that in the last section, we compute the link score between a tag and a concept (See Table I). It can be applied to estimate $P(c_z|t_y)$, as well. $P(t_y|S_i \cup S_j)$ can be estimated as the relative frequency of t_y in $S_i \bigcup S_j$.

$$P(c_z|t_y) = \frac{w(t_i, c_j)}{\sum_{c' \in \mathcal{C}} w(t_i, c')}$$

Iterating all tags in $\mathcal{W}$ is costly. Instead, in the computation, we only use tags in $\mathcal{T}$ to estimate Equation 3.1. Based on the analysis and experiments in [36], such simplification does not degrade the accuracy significantly while save a lot of computation. In addition, we normalize the KL value by the total tag number in the subset.

Given a crawled resource with a tag set $\mathcal{T}$, Algorithm 3.1 illustrates how we group the tags into disjoint subsets. Initially, $|\mathcal{T}|$ subsets are generated with each subset only containing one tag (line 2-4) and a KL-Matrix is computed to record the gain value (introduced later) between any two subsets (line 5-8). Then, we iteratively combine the subsets with current maximal gain until reaching the threshold (line 9-23). When a new subset is created, we replace the old entries in the matrix with new ones (line 16-21). In this way, the gain is incrementally updated to support the next iteration of clustering.

Algorithm 3.1: Wikipedia based Tag Clustering

Input: Tag set $\mathcal{T}$ of the input resource
Output: Subsets of $\mathcal{T}$

AllSubset $result = \emptyset$;
foreach $t_i \in T$ **do**
    Subset S_i = new Subset(t_i) ;
    $result$.add(S_i);
foreach i= *1 to* $result$.*size* **do**
    foreach j= *1 to* $result$.*size* **do**
        if $i \neq j$ **then**
            $M_{i,j} = Gain(S_i, S_j)$ //KL matrix;
while *true* **do**
    Double max=0, int a=0, int b=0 ;
    foreach i= *1 to* $result$.*size* **do**
        foreach j= *1 to* $result$.*size* **do**
            if $M_{i,j} > max$ **then**
                $max = M_{i,j}$;
                $a = i, b = j$;
    if $M_{i,j} > th$ **then**
        Subset $S_{new} = S_a \cup S_b$;
        $result$.remove(S_a), $result$.remove(S_b) ;
        $result$.add(S_{new});
        update M by removing the rows and columns involving S_a and S_b;
        add in a new row and column in M for S_{new} to record the gain of S_{new} to all other subsets ;
    else
        break;
return $result$;

In the experiment, we discover that if one of the child subsets gets a high KL score, then the score of the new merged subset tends to be high, as well. In order to capture the benefit of forming a new subset more accurately, we use the gain of KL score as the merging criteria.

$$\begin{aligned} Gain(S_i, S_j) &= KL(S_i, S_j) - KL(S_{ii}, S_{ij}) \\ &+ KL(S_i, S_j) - KL(S_{ji}, S_{jj}) \end{aligned}$$

$S_{ii}, S_{ij}, S_{ji}, S_{jj}$ are the respectively child subsets of S_i and S_j. The value of the threshold th should be set adaptively based on the tag distribution. Our experience shows that using a small number of samples is enough for estimating a good threshold.

The generated subsets are then sorted by their weights, which are computed as follows:

$$W(S_i) = |S_i| * \sum_{t_x \in \mathcal{T}} P(t_x|S_i) \log \frac{P(t_x|S_i)}{P(t_x|\mathcal{W})} \tag{3.3}$$

The weight definition combines the size of the subset and the tag distribution. It can be evaluated in the same way as Equation 3.1. In our current implementation, we assume that each resource only has one core topic. Therefore, we just keep the subset with the highest weight. In other words, we prune the unimportant tags from the tag set $\mathcal{T}$.

3.2.2 Concept Mapping

Our selected tags are used to discover correlated concepts of Wikipedia and establish the mappings between the resource and concepts. We observe that most tags are linked to more than one concept, which can be classified into different categories. For instance, the tag *Marina Bay* is linked to a few concepts such as "Marina Bay MRT", "Marina Bay Sands" (the casino), "Marina Bay Financial Centre" and "Marina Bay Singapore". Supposing the image is taken for the Merlion at Marina Bay, only the last concept is the correct match while the other concepts, if applied in search, will cause ambiguities.

To address this problem, we exploit the context of tags. For a resource, all its tags in $\mathcal{T}$ after pruning are considered as context tags to each other. If an image is tagged with *Merlion, Singapore, Nikon* and *Marina Bay*, we can infer that *Marina Bay* refers to the concept "Marina Bay Singapore". For a tag $t \in \mathcal{T}$, its context tag set is $\mathcal{T} - \{t\}$. Let $C(t)$ and $\mathcal{C}$ denote all concepts linked by t and the whole concept space, respectively. We have $C(t) = \{c|w(t \rightsquigarrow c) > 0 \wedge c \in \mathcal{C}\}$. The similarity between concepts in $C(t)$ and t can be estimated as:

$$s(c,t) = P(t|c)P(\mathcal{T} - \{t\}|c) = P(t|c)\Pi_{t_i \in \mathcal{T} - \{t\}} P(t_i|c) \tag{3.4}$$

$P(t|c)$ can be computed as $\frac{w(t \rightsquigarrow c)}{\sum_{t_x \in \mathcal{W}} w(t_x \rightsquigarrow c)}$. Otherwise, t_i does not have explicit connection with c ($w(t_i \rightsquigarrow c)$). In this case, we use the concept correlations to discover the hidden semantic links between tags and concepts. Formally, we expand $P(t_i|c)$ as

$$P(t_i|c) = \sum_{c_j \in C(\mathcal{T})} \frac{P(t_i|c_j)P(c_j|c)}{d_c(c_j, c)} \tag{3.5}$$

where $P(t_i|c_j)$ is computed as Equation 3.2, $P(c_j|c)$ is the correlation between c_j and c (see Table I). $d_c(c_j, c)$ is the semantic distance between c_j and c. Combining Equation

Figure 3.3: A Query Image and Its Top-4 Intra-domain Similar Images

3.4 and 3.5, we can compute the weights of concepts for each tag. Finally, the candidate concept set of a resource consists of concepts with the highest score of each tag: $C(\mathcal{T}) = \bigcup_{t \in \mathcal{T}} \{c | \max s(c,t) \wedge w(t_i \rightsquigarrow c_j)\}$

Finally, the uniform concept vector of the resource can be generated by iterating all concepts in $C(\mathcal{T})$.

1. If c_j is not in $C(\mathcal{T})$, w_j is set to 0.
2. Otherwise, $w_j = \sum_{t_i \in \mathcal{T}} s(c_j, t_i) * W(S)$. $W(S)$ is the weight of the generated tag subset of $\mathcal{T}$ which can be computed as in Equation 3.3.

In the offline processing, we compute the concept vectors for all resources that we crawled before and store them in our database. Thus, the various modal resources are transformed to the uniform representations which facilitate for future operations.

3.3 Cross Domain Search

Our cross domain search accepts various types of queries. The user can submit typical keyword queries, or he can upload an image or document as the query. The difference between the cross domain search and conventional search service is that the user can issue a query to any specific domain, and the cross domain search can return results from different domains. For example, the mobile user can take a photo of a building and submit the image as a query to find the documents, videos and other images associated with the building.

In our system, the query is transformed into the uniform concept vector, and we retrieve the crawled resources with similar concepts as the results. However, different from

crawled resources, which are tagged by users, the query normally does not come along with any tag. We cannot exploit the tag-concept links as in the last section. Therefore, we plan to build the concept vector of a query by leveraging its **top-K Homogeneous Resources** which are already stored in the database.

Definition 3.5. *Top-K Homogeneous Resources (THR): Given a query Q, its THR should satisfy the following three requirements: $\forall r \in Q.THR$*
1) Both r and Q are in the same domain.
2) r is already represented by a uniform concept vector.
3) r is one of the top-k similar resources of Q according to a similarity function.

Next, we first introduce the way of extracting THR of a query.

3.3.1 Intra-Domain Search

For query q and resource r_i in the same domain $Domain_j$, the query processor computes their similarities via the similarity function $Sim(q, r_i)$. Only the resources with high similarities are returned as the results. In this section, we use text documents and images as our examples to show the idea of intra-domain search. As a matter of fact, other domains can be easily integrated if the similarity function has been defined.

- *Document:* In the domain of text documents, each document is represented as a word vector, $v(r_i) = (t_1, t_2, ..., t_n)$. Each dimension of the vector refers to a term, and its value denotes how important the term is. In this work, $tf - idf$ is used as our metric to measure the term weights. Therefore, $t_j = tf_{ij} \times idf_j$. The query in this domain is also transformed into a word vector $v(q)$ and similarity is computed as the cosine distance.
$$Sim(q, d_i) = \frac{v(d_i) \times v(q)}{|v(d_i)||v(q)|}$$
- *Image:* In the domain of image, each image is represented as a feature vector, $v(r_i) = (f_1, f_2, ..., f_n)$. Each dimension of the vector represents an image feature, such as colour, texture and shape. These features are proven to be effective in many applications [24]. In our system, the visual features of the image are viewed as a distribution in the visual space. Thus, the similarity between images and query images is captured by the KL divergence.
$$Sim(q, d_i) = \sum_{j=1}^{n} v(q).f_j \times \log \frac{v(q).f_j}{v(d_i).f_j}$$

Parameter K of THR may affect the recall and precision of the results, and this will be studied in our experiments.

3.3.2 Building Uniform Concept Vector for Queries

One straightforward way of building the vector is to combine the concept vectors of its THR. Let THR be the candidate resources of query q. We generate the concept vector for q by aggregating the vectors of the candidate resources:

$$v_q = \sum_{r_i \in THR} v_i$$

For a concept c_j, its weight for q is

$$w(q)_j = \sum_{d_i \in THR} v_i[w_j]$$

where $v_i[w_j]$ is the jth weight of d_i's concept vector.

However, we should note that not all the concepts existing in the concept vectors of its THR are the proper description of the query. For example, Figure 3.3 shows a query image and its top-4 homogeneous resources. All four candidates are very similar to the query image based on their image features. However, in fact, two of them (a and c) are not related to the query image and will bring misleading concepts. Using them as candidate resources will generate noisy concepts. In addition, the above naive ranking approach fails to consider the correlations between concepts and assumes that all candidate resources are equally valuable. This may degrade the precision of results because the false positives (such as a and c in Figure 3.3) are introduced into the search process. To improve the quality of the vector, we adopt a more sophisticated ranking approach. The intuition is that different from the noisy candidate resources, the good results always have correlated concepts (the concepts of the query).

Our ranking approach considers three factors:

1. *Resource Importance*: Concepts from different candidate resources should not be considered equally. For a concept c_j from resource r_i, we normalize c_j's weight by $Sim(q, r_i)$, the similarity between the query and the resource.

2. *Concept Importance*: Each concept in Wikipedia links to some other concepts. If concept c_i exists in the article of concept c_j, we say c_j is linked with c_i. Usually the general concepts will connect to more concepts than the specific concepts. We propose to compute a concept's generality as follows:

$$g(c) = \frac{1}{log\frac{|\mathcal{C}|}{|links(c)|} + \lambda}.$$

where $|\mathcal{C}|$ denotes the total number of concepts in Wikipedia, $|links(c)|$ indicates the number of concepts linking with c and λ is a parameter to smooth the function. A larger $g(c)$ implies a more general concept, which is less distinguished in search.

3. *Concept Correlation* : We exploit the correlations between concepts to find the hidden semantics. Given a concept c which does not appear in resource r_i from THR, we can measure its weight via its correlations with other concepts in r_i. Specifically, for resource $r_i \in THR$,

$$s(c, r_i) = \sum_{c_j \in r_i.concept} P(c|c_j)v_i[w_j]$$

where $P(c|c_i)$ is the correlation in Table I and $v_i[w_j]$ is the jth weight of resource r_i's concept vector which corresponding to c_j.

Combining the above factors, we adjust the scores of concepts in the candidate resources. For query q, the new score of c is computed as:

$$w(c, q) = \frac{\sum_{d_i \in THR} s(c, d_i)Sim(q, d_i)}{g(c)} \tag{3.6}$$

In this way, we can generate the uniform concept vector for q and use it to rank the resources of different domains.

3.3.3 Resource Search

To facilitate the search, all concept weights are normalized into the range of $[0, 1]$. The uniform concept vector can be considered as an n-dimensional point, where n is the number of concepts. Although there may be millions of concepts, for each resource, only a small portion of concepts are involved. Given a query q and its concept vector $v_q = (w_1, w_2, ..., w_n)$, we only need to retrieve the resources, which share at least one

concept with q. For this purpose, an inverted index $(cid, DList)$ is built. cid is the concept ID and the $DList$ is a set of resources, satisfying that

$$\forall r_i \in DList \rightarrow v_i[cid] > 0$$

Namely, the concept vectors of resources in $DList$ have a nonzero weight for concept cid. Let v_q and v_i be the concept vector of query q and resource r_i, respectively. Their similarity is estimated by combining the cosine distance of the concept vectors, and the intra-domain similarity:

$$Score(q, r_i) = \frac{v_q \times v_i}{|v_q||v_i|} + Sim(q, r_i)$$

If q and r_i are in different domains, $Sim(q, r_i)$ is set to 0. Otherwise, it can be computed by the intra-domain similarity function.

3.4 Experimental Study

We have implemented the cross domain search framework [57] based on the proposed method. In this section, we evaluate its performance with real data set from UGC websites. In particular, we collect images from Flickr and documents from Delicious as the experimental data. After removing the duplicated contents, $136k$ images and $114k$ documents are stored in the repository. Those images/documents are considered as the cross domain resources in our system. These resources could be classified into different categories, such as food, landmarks, sports, air planes and many others. The English version of Wikipedia, which is released on $2008.07.24$, is employed as our knowledge base. It contains around $2.7M$ concepts, covering almost every domain. We summarize the data set statistics in Table 3.2.

In the following subsections, we perform a comprehensive study on our designed strategies. All of our experiments are conducted on a Linux server with Quad-Core AMD Opteron(tm) Processor 8356, 128GB memory and running RHEL 4.7AS.

Table 3.2: Cross Domain Resource Data Statistics

	Resource Number	**Unique Tag Number**	**Total Tag Number**
Flickr	135,962	120,097	1,290,511
Delicious	114,085	133,943	4,006,636

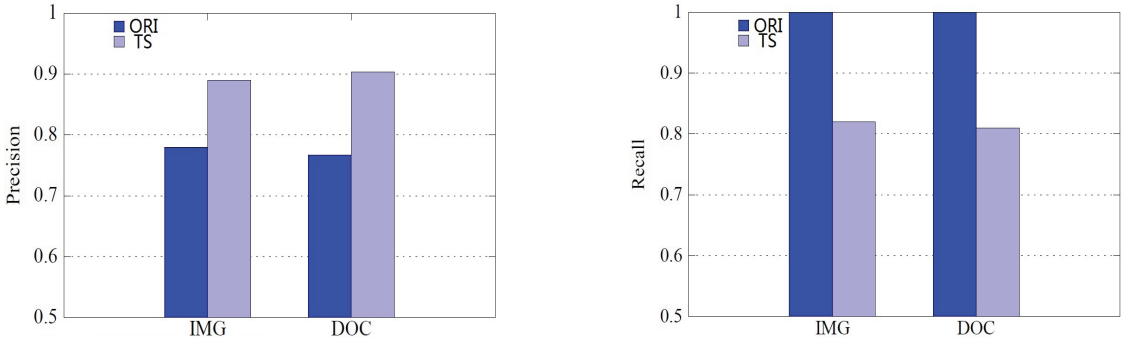


Figure 3.4: Tag Selection Performance on TR

3.4.1 Evaluation of cross-domain concept links

In our system, each resource is linked to a set of Wikipedia concepts with a two-step process. In the first step, we cluster the tags and select the key tags to represent the resource. In the second step, based on the concept correlations, we select a concept with the highest weight for each left tag. In this experiment, a test resource set *TR* is generated by randomly choosing 25 images (referred as *IMG*) and 25 documents (referred as *DOC*) from the underlying data repository. The average number of tags for resources in *IMG* and *DOC* are 7.7 and 9.4 respectively.

1) Tag Selection: To measure the quality of tags selected by our clustering algorithm, we define two evaluation metrics: *precision* and *recall*. The $precision$ is computed as $\frac{r}{l}$, where l is the total number of tags associated with the resource and r is the number of relevant tags. $precision$ is used to measure the effectiveness of our algorithm in terms of removing unimportant tags. On the other hand, the $recall$ is computed as $\frac{h}{r}$, where r is defined as above and h is the number of relevant tags selected by our algorithm. A high $recall$ indicates that our algorithm can retain most relevant tags.

Table 3.3: Examples of Tag Selection

Original Tags	**Processed by TS**
Singapore, Grand Prix, 1, F1, FIA Motorsport, Racing	F1, Racing, Motorsport, FIA
Singapore, Merlion, Marina Nikon, Photograph	Singapore, Merlion Marina
3d, artist, colour, colourwallah exhibition, gallery, london, sculpture	sculpture, exhibition artist

We illustrate how these two scores are computed using the examples in Table 3.3. The

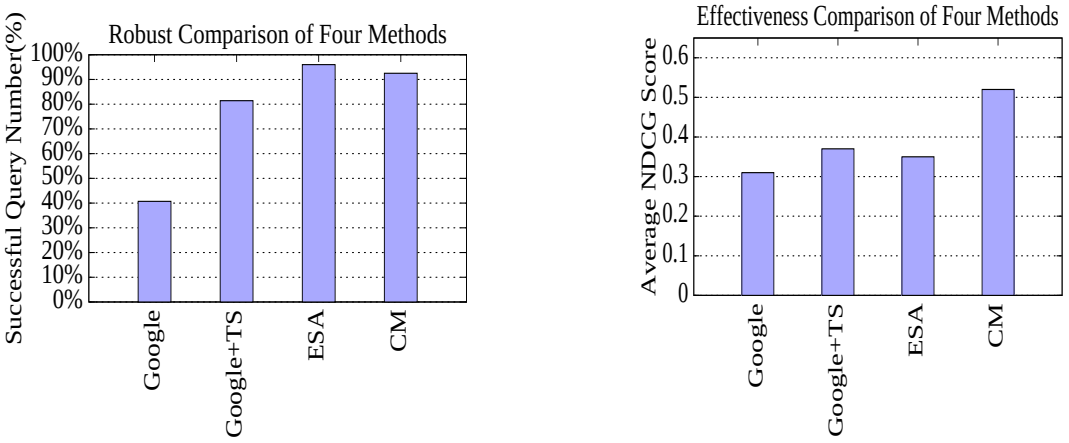


Figure 3.5: Comparison of Four Methods

first row is about an image, which describes a F1 racing car in Singapore Grand Prix. The left column shows the original tags annotated by the user while the right column is the tags returned by our algorithm. As "Singapore, Grand Prix, F1, FIA, Racing, Motorsport" are the relevant tags and "1" is not, the precision of original tagging (ORI) is $\frac{6}{7}$. Correspondingly, the precision of the tags which are processed by our tag selection algorithm (TS) becomes $\frac{4}{4}$ since all 4 selected tags are relevant. We assume ORI contains all the relevant tags. Thus, the recall of the ORI is always 1. On the contrary, we can compute the recall of TS as $\frac{4}{6}$.

The comparison result of ORI and TS is shown in Figure 3.4. The high precision and recall indicate that TS is capable of removing most irrelevant tags and keep the relevant tags as many as possible. The reason that TS achieves such a high precision is because of the usage of clustering. If we are able to identify the correct cluster, with a high probability, the tags in the cluster are all high-quality tags for the resource.

For the recall perspective, although TS falsely removes some relevant tags, the average recall, 0.8, is still acceptable. In most cases, the false negatives are caused by the tags describing the context of the resource, e.g., "Singapore" in the above example. It is too general to be categorized into a cluster with other tags. Another reason is that we only keep the tags in the highest weighted cluster while there are some relevant tags exist in other clusters. In the future, we will study how to set the cluster number adaptively.

2) Concept Mapping: In this experiment, we show the effectiveness of using uniform concept vectors to represent the resources. Specifically, we compare our concept mapping method (*CM*) with three other concept generation methods, including *Google*, *Google+TS*

and *ESA*. In *Google* method, we query Google by the tags of a resource in *TR* and restrict the search scope in Wikipedia.org. The returned article titles are used as the resource's Wikipedia concepts. This method is used to verify whether the search engine method still works in our cross domain scenario. In *Google+TS* method, we first process the queries by our *TS* algorithm and then search in *Google*. By doing this, we examine whether the performance will be improved if we apply our tag selection on the tags. *ESA* [31] is the state of the art technique introduced before.

Robustness : Figure 3.4.1 shows the percentage of queries that have valid results in each method. We find that *Google* only returns valid results for about 40.7% queries while the rest of queries gets no result at all. The reason is that Google cannot effectively handle the queries which contain ambiguous keywords or too long. However, after enhancing by our tag selection techniques, the *Google+TS* scheme becomes more robust since the spam tags are pruned. The best results are observed from the CM, and ESA approach, which can process more than 90% queries.

Effectiveness : to measure the effectiveness of different schemes, we adopt the Normalized Discounted Cumulative Gain (NDCG) metric [44]. In this metric, the relevance of returned results is labelled as one of the five levels (1-5). The average NDCG score for the query set is computed as follows:

$$NDCG(Q,k) = \frac{\sum_{j=1}^{Q} Z_{kj} \sum_{i=1}^{k} \frac{(2^{r(j,i)}-1)}{log(1+i)}}{|Q|}$$

where Z_{kj} is a normalization constant so that the optimal NDCG is 1, and $r(j,i)$ is the relevance level of the ith concept (tag) in query j. This metric takes both relevancy, and ranking order into consideration, and is widely used to measure the quality of the results. To compute the NDCG score, ten students from the CS department are volunteered to give scores for each query.

Apparently, CM approach outperforms all the other methods as shown in Figure 3.4.1. This is because the concepts in CM are generated for each tag. Therefore, fewer noisy concepts will be introduced. Note that, one limitation of CM is that when a phrase is falsely segmented into separated parts, e.g., from "Paris Hilton" to "Paris", and "Hilton", we are unable to find the correct concept ("Paris Hilton", the celebrity). This also explains why our method performs worse in *DOC* than in *IMG*. As in Delicious, the false segmentation happens more frequently. In addition, the comparison of $Google$ and $Google + TS$ demonstrates the usefulness of our *CM* method in another perspective. It illustrates that we can indeed remove the noisy tags from the resources.

3.4.2 Evaluation of Cross Domain Search

In this section, we compare our concept-based retrieval (CBS) with the tag-based retrieval (TBS & $TBSR$). TBS adopts the similar idea as CBS but based on the original tags. The difference between TBS and $TBSR$ is that $TBSR$ works on tags which are processed by our tag selection algorithm. We expect to further verify the effectiveness of the algorithm, and examine whether representing resources with Wikipedia concepts could improve the performance. The resources in TR are used as queries. In the test, we remove the tags for the query resource; hence the query is just the pure image or document. As mentioned in Section 3.3, we perform the cross domain search in two steps: building the concept vector for the query and performing the resource retrieval in the repository.Before evaluating the two steps, we first investigate the effect of K in THR.

1)THR Selection : to generate the uniform concept vector of a query, we exploit its Top-K homogeneous resources. The effect of different K values is shown in Figure 3.6. The y axis represents the NDCG score of the generated concepts for different K. In the diagrams, both of the two data sets achieve their highest score when $K = 50$. The reason is that if K is set to a small value (e.g., 20), many related concepts cannot be discovered due to the small number of intra-domain resources. On the other hand, when K is set to a large value (e.g., 100), noisy concepts are introduced, and they may increase each others' weights via the correlations. Another interesting observation is that the scores of DOC are higher than those of IMG. It is because the intra-similarity function between documents is much more accurate than that of the images. Therefore, more relevant concepts can be retrieved in the document setting. The third finding is that the distance between scores of IMG is smaller than that of DOC. After checking the data, we find that the distance of intra-similar score between the image query and its candidates are much greater than the scores between text query and its similar documents. This illustrates that the image candidates ranked in the lower positions do not have much influence as that of DOC.

2) Uniform Concept Vector Building for Queries : We compare the relevancy of concepts (tags) generated by *CBS*, *TBS*, *TBSR* and the fixed annotation-based cross-media relevance model (*FACMRM*) [45]. *FACMRM* is an automatic image annotation method. By using a training set of annotated images, FACMRM learns the joint distribution of images and words. Then fixed length annotations can be generated by using the top k words to annotate the images. Here, we adapt the FACMRM idea as a method of generating tags for a query. In the evaluation, we choose TR as the query set and NDCG as the measurement metric as before.

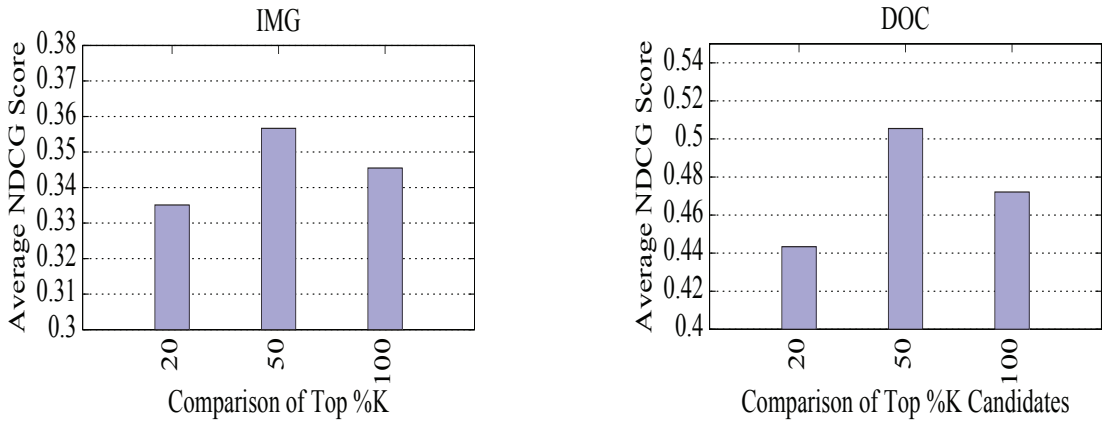


Figure 3.6: Top K Comparison

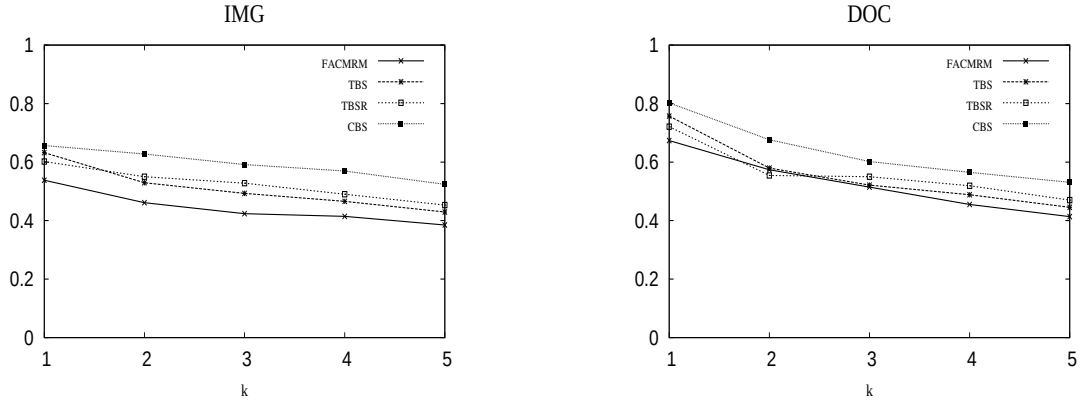


Figure 3.7: Average NDCG Comparison of Concept (Tag) Retrieval

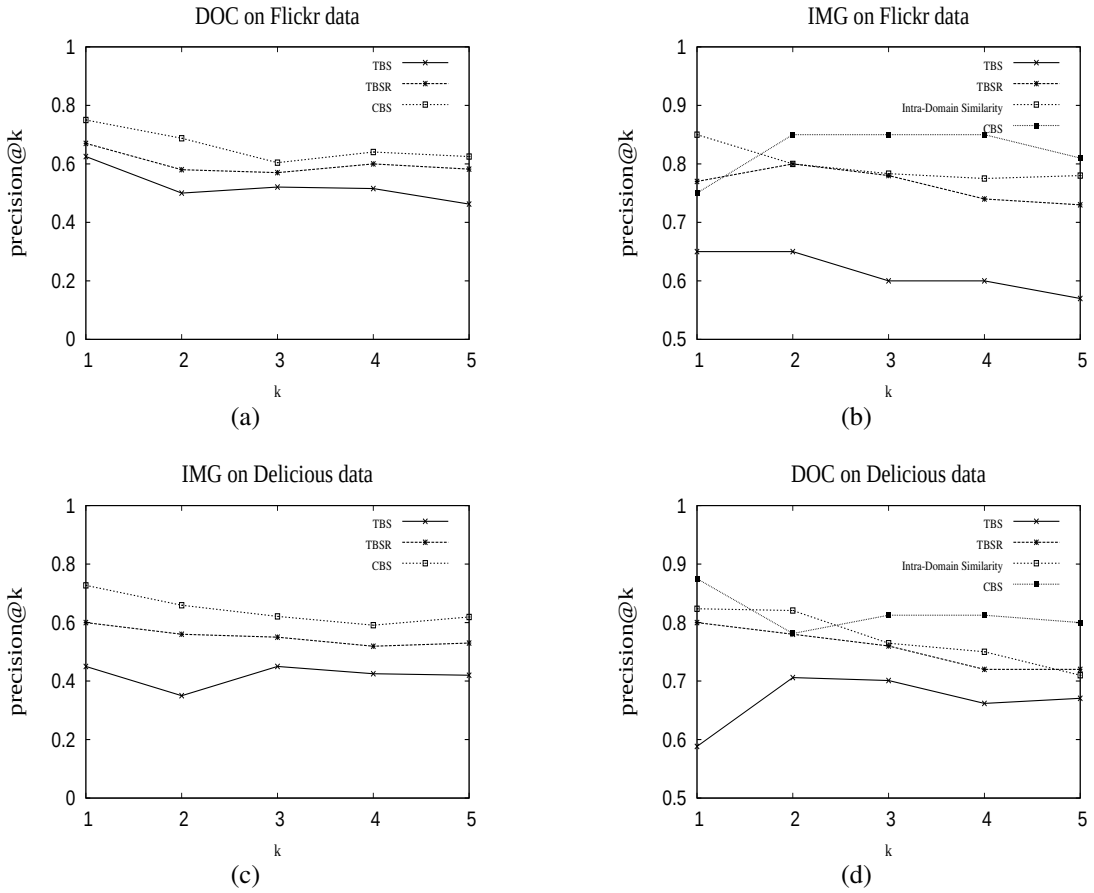


Figure 3.8: Precision@k Comparison in Different Settings

Figure 3.7 illustrates the NDCG score comparison between the four methods. In the four methods, the highest score is observed when only the first concept (tag) is used. This indicates that the most relevant concept (tag) is always ranked in the top. Among the tag-based methods, TBS performs better when K is small. We analyse the reason as that sometimes, $TBSR$ will falsely recognize the useful tags as noise ones. Thus, the performance will be hurt. In CBS, we add the correlation factor into the concept ranking. We believe this is an influential action which leads to its best results. We manually check the ranked concepts (tags) and find that our concept based method tends to reduce the bias which may exist in the underlying repository. For example, "Merlion" is tagged much less than "Singapore" in the repository. Thus, "Singapore" has a larger possibility to appear in the *THR* of a query. However, ranking "Singapore" higher does not help the search much as it is too general. Our method can overcome this problem, and rank "Merlion" higher.

The average scores for images are lower than that of documents. The difference is caused by the semantic gap between low level features and high level semantics of images. The images, which are similar in vision, may not be similar in semantics. Simply relying on

visual similarity will introduce noisy concepts. Compared to images, it is more credible to capture the semantic similarities between documents.

3) Cross Domain Search : We perform a set of experiments to measure the quality of cross domain search. The results are evaluated in four categories: searching images by an image, searching images by a document, searching documents by a document and searching documents by an image. For each query, we select the top-5 concepts (tags) to build the concept vectors, and then retrieve the relevant images and documents respectively.

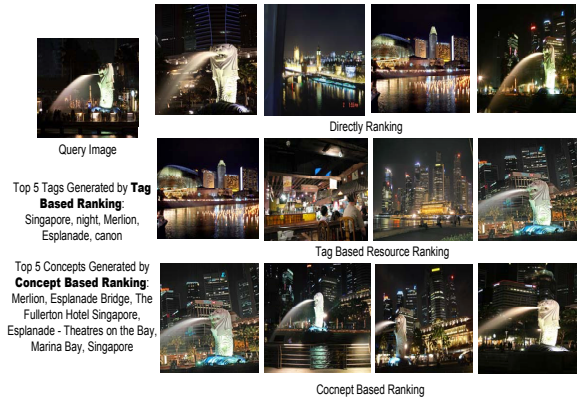
Figure 3.8 presents the comparison results of the $precision@k$ (top k returned results). We also present some examples in Figure 3.9 to better illustrate our results. From the figures and examples, it is clear to show that *CBS* does work and performs better than the other methods. The reason is that by consistently representing the resources and queries in uniform concept vectors, we can capture the semantics between resources more precisely. Another reason is that concepts can receive a more reasonable weight than the tags. The weights of the concepts of queries play important roles in retrieving the correct resources. We find that even for the same concept set, even when their weights vary a bit, the final results will change significantly.

We also compare our methods with the intra-domain similarity function when the query and results are in the same domain. From Figure 3.8(b) and 3.8(d), we observe that adding one more step is a double-edged sword. We analyse the reason as follows. If the tags are not assigned correctly, errors will be introduced and propagated in each step. Finally, the results will be degraded, such as TBS. However, if we can assign and weight the intermediate concept properly, it does help to improve the performance.

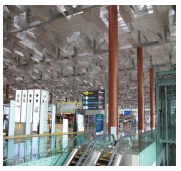
Another trend observed from the above figures is that the precision of cross domain search (Figure 3.8(a), 3.8(c)) is lower than that of intra-domain search (Figure 3.8(b), 3.8(d)). Except the inherent difficulty in cross domain search, it is partially caused by the incompleteness of the underlying resource repository. For example, for some image queries, we have a few or none relevant document resources in our repository. We need to enlarge our repository next.

3.5 Summary

In this chapter, we propose a brand new cross domain framework for web-based applications, extending the existing annotation method by linking the resources to concepts in



(a)



TBS	TBSR	CBS
http://www.zoo.com.sg/Index.com	http://www.asiaone.com.sg/	http://www.airporthotel.com.sg
http://www.visitsingapore.com/publish/stbportal/zh/home.html	http://www.mom.gov.sg/publish/momportal/en/home.html	http://delhi.bigcities.org/airport.html
http://www.streetdirectory.com/	http://www.airportotel.com.sg/index.htm	http://www.heathrowairport.com/
http://www.thingstodoingapore.com	http://www.visitsingapore.com/publish/stbportal/zh/home.html	http://wo61.healthyounger.com/singaporechangiairport.html
http://www.changiairport.com.sg/changi/en	http://www.changi.airport.com.sg/	http://www.heathrowairport.com/portal/site/heathrow/

(b)

(c)

Query URL: http://www.gp2006.com/home

Top 5 Tags Generated by TBS: Automobile , Auto racing , Video game, Formula One ,Sport

Top 5 Tags Generated by TBSR: Sports, cars, moto, F1, racing

Top 5 Concepts Generated by CBS: Automobile, Auto racing, Porsche, Formula

Directly Ranking	TBS	TBSR	CBS
http://clubf1.net/new/index.php	http://www.tmsf.jp/	http://www.eurosport.fr/formule1/	http://www.f1.com/
http://www.tmsf.jp/	http://www.americanlemans.com/news/Article.aspx?ID=2691	http://www.formel1.de/index.php?id=8	http://blogs.iht.com/tribtalk/sports/f1/index.php
http://video.google.com/videoplay?docid=6756084931270105691	http://www.cbracing.com/	http://www.lsbu.ac.uk/sports/sportshall.htm	http://www.autosport.com/gallery/
http://motorsportlegend.com/	http://au34.motoristsmico.com/automobileracing.html	http://www.zdkf1.com/login.php	http://www.formula1.com/news/
http://www.ferrariworld.com/	http://blogs.iht.com/tribtalk/sports/f1/index.php	http://www.zdkf1.com/login.php	http://www.formula1.com/gallery

(d)

Figure 3.9: Four examples of Cross Domain Search

Wikipedia. Our framework fully utilizes the Wikipedia concepts with their well-organized contents and high-quality links. Our framework exhibits high extensibility and flexibility on the processing of cross domain search, which only depends on the correlation between the resources and the Wikipedia concepts. Our experiment results show that our proposal dramatically improves the search quality, as well as presents considerable potentials on the enhancement of UGC applications.

In summary, the main contributions of this chapter includes:

- We present a general framework to integrate resources from different UGC websites and support cross domain search.
- We provide several methodologies which can be applied to many other applications, such as the spam tag detection, topic clustering, query expansion and automatic image annotation.

- We conduct a comprehensive performance study to validate our techniques.

CHAPTER 4

TWITTER IN A PERSONALIZED KNOWLEDGE VIEW

In the previous chapter, we construct references for resources from different domains. Consequently, cross domain search is supported across UGC websites. Given the rich resources and their references, the way of organizing and presenting them remains a problem. Currently, the resources are usually organized by their freshness with the latest one ranking in the top. Figure 4.1 gives a snapshot of the Twitter interface. In the interface, all the incoming tweets are listed based on the temporal dimension. Although such an organization looks clean and straightforward, it brings great troubles for a user who uses Twitter to gain knowledge about a hot topic (e.g., cloud computing and big data), an industry (e.g., mobile technology) or even an astonishing product (e.g., Google Glasses). The user has to go through all the resources in order to find the interesting/useful information. It takes considerable efforts to filter these tweets manually. Hence, users can be easily immersed in the mass data yet miss the useful information. This dilemma is usually referred as "information overflow" problem, which remarkably exists in different UGC websites. In this part, we take Twitter as an example to show how we approach the problem. In the literature, to address the problem, some researchers proposed tweet ranking to sort tweets considering user interests [35, 15]. The basic idea is that a tweet will be displayed in a higher position if it is more relevant to a user's profile. The ranking is performed based

Figure 4.1: A Snapshot of User Interface in Twitter

on the text information without consideration of semantics among tweets. In addition, as stated in [62], these ranking methods may not work well in Twitter due to the noisy and short traits of tweets.

In this chapter, except the temporal dimension, we propose to add a knowledge dimension to organize and present resources in UGC websites. As a start, we take Twitter as an example to illustrate our methodology.

There are two challenges in building the knowledge view of Twitter.

1. *Knowledge Organization :* The first challenge is how can we add knowledge or semantics for tweets since they are short and noisy. What the view looks like is another problem.

2. *Knowledge Personalization :* In order to provide a personalized knowledge view, the system needs to take both user interests and knowledge into account. The second challenge is how to combine them together.

To address above issues, we use Wikipedia as the knowledge base and build a knowledge graph by extracting its hierarchical structure. For the tweet stream, each of them will be mapped into the nodes of the knowledge graph. Then a number of trees are built by connecting the mapped nodes. These subtrees are displayed with labels on each level and

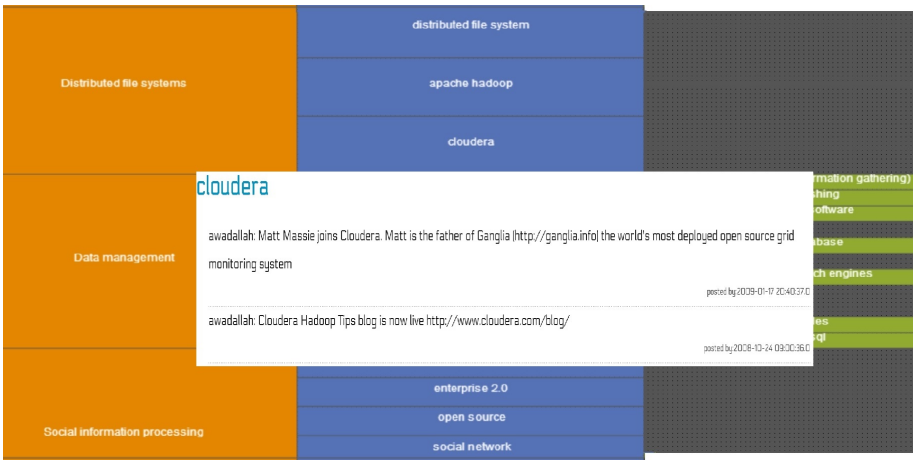


Figure 4.2: Knowledge View for a User

ranked based on a tree kernel model. In this way, we add a knowledge dimension to the tweet stream. To present a clear idea of the new interface, we give an example in Figure 4.2 which shows the interface for user "Alon Y. Halevy". Alon Y. Halevy is a famous researcher with the interests on data integration, personal information management and etc. We organize his tweet stream in several lines, each line represents a hierarchy structure of the knowledge. The topic labels on the left are more general than those on the right. Users can quickly identify the related tweets by clicking on the node label. Then related tweets will be popped up. As shown in the Figure, we display the information about "distributed file system" or "data management" in the higher position which fits for his research interests. In this new interface, we not only illustrate the tweets that are related but also explain why they are related.

The rest of the chapter is organized as follows. In Section 4.1, we describe the problem statement and the workflow of our system. We explain the details of each step in the following sections. Section 4.5 illustrates the evaluation methodologies, experiments and results of our framework. Before make the conclusion in Section 4.7, we discuss the interface in Section 4.6.

4.1 Problem Statement

Twitter is a social media platform where users can follow other people to acquire the information that they are interested in. The following relationships, denoted by $\Longrightarrow$, form a directed graph. We use $F(u)$ to denote the group of people that a user u is following so that

$$F(u) = \{v | u \Longrightarrow v\}.$$

$F(u)$ could generate a considerable amount of tweets every day and these tweets are ordered by the timestamp to emphasize the real-time property of Twitter. In this way, users can easily identify which tweets are the latest. We use $T(u)$ to represent the tweet stream for user u to consume. Since the size of $T(u)$ could be beyond user's consume capacity, interesting tweets may be missed. In this chapter, we propose an alternative way to organize $T(u)$ for the user so that useful and interesting information will be promoted to reduce the probability of missing. We will formally define the problem of personalized knowledge extraction, organization and visualization given the tweet stream $T(u)$.

In order to discover and extract semantic information from tweets, we utilize Wikipedia as the knowledge base. There are two reasons that make it as our choice. First, the articles in Wikipedia are of high quality. Ambiguity (single term with multiple different meanings) and synonym (multiple terms have similar meanings) are well solved in Wikipedia. For example, given a word "Apple" in a tweet, from the disambiguation page of "Apple" [1], we can obtain all the possible meanings of this term, including companies, films, music, etc. Then we can choose the most appropriate one depending on the context. In Wikipedia, each article describes a unique topic and is associated with a concept as the semantic attribute. In this chapter, we use the article title, which is a succinct and well-formed phrase, to represent the concept. The second reason to use Wikipedia is that there is ontology embedded in the articles to capture relationships between different concepts. One of such examples is the category, which provides a hierarchy structure to organize the semantic. In the following, we use C to denote the concept set and $\mathbb{C}$ to represent the category set. For any concept c from C, we can find at least a category $\mathtt{c} \in \mathbb{C}$ such that c belongs to category $\mathtt{c}$, denoted by $c \rightarrowtail \mathtt{c}$. Thereafter, we can define the knowledge graph based on the concept and category.

Definition 4.1. *Knowledge Graph*
A knowledge graph organizes the concepts and categories from Wikipedia in a directed acyclic graph $G = (V, E)$ with a root node R, where $V = \{v | v \in C \text{ or } v \in \mathbb{C}\}$ and for any edge $(u, v) \in E$, we have $u \rightarrowtail v$. The root node R in Wikipedia is `Category:Fundamental`.

Given the knowledge graph, we can clear the noisy terms in tweets and then associate the remaining meaningful terms to Wikipedia concepts. Since one tweet can be assigned to multiple concepts and there could be thousands of tweets in $T(u)$, a great number of concepts may be extracted from $T(u)$. How to effectively and efficiently organize these concepts is a challenging problem. In this chapter, we define it as a knowledge

[1] http://en.wikipedia.org/wiki/Apple_%28disambiguation%29

organization problem which connects the related concepts in a number of knowledge trees derived from the knowledge graph. A knowledge tree is a subtree of the knowledge graph and its leaf nodes are required to be concept nodes.

Definition 4.2. *Knowledge Subtree*
A Knowledge Subtree is defined as $\mathcal{S} =< V_T, E_T >$*, where* $V_T \subset V_G$*,* $E_T \subset E_G$ *and* $V_{leaf} \in C$*.*

The knowledge organization problem can be formally defined as follows:

Definition 4.3. *Knowledge Organization*
Given a collection of weighted concept nodes $C' \subset C$ *extracted from* $T(u)$*, the knowledge organization problem finds* k *sub-trees* $\mathbb{S} = \{S_1, S_2, ..., S_k\}$ *covering all the nodes in* C' *and with the minimum weight under a certain weighting function.*

To support the personalized recommendation, we assign a relevance score to each knowledge tree with respect to the user. Hence, the knowledge personalization problem is essentially a ranking problem to order the resulted knowledge trees so that the most relevant subtrees are displayed on the top to avoid missing by the user.

Definition 4.4. *Knowledge Personalization*
Given a collection of knowledge subtrees $\mathbb{S} = \{S_1, S_2, ..., S_k\}$ *and a user model* $\mathbb{M}$*, the knowledge personalization corresponds to a ranking function* $F(S_i, \mathbb{M}) \rightarrow R$ *such that* $R_{S_i} > R_{S_j}$ *if* S_i *is more interesting to the user than* S_j*.*

Figure 4.3 illustrates the framework of our system. Given a user, the system accepts two tweet streams. One is the stream containing tweets that are from followings of a user. We call it as the incoming tweet stream. The other contains tweets composed by the user and are spread to his followers. We call it outgoing stream or personal stream. It will be used as a raw data source to model the personal interests. In the back-end of the system, we construct a knowledge graph from Wikipedia and use it as our knowledge base. When the new tweet flow arrives, they will be first mapped to the concept nodes in the knowledge graph. Then, similar to spanning tree construction, we will find the best connected components in the knowledge graph that cover all the concept nodes. We prove that it is a NP-Hard problem to find the best solution among all the possible organizations. Hence, efficiency and effectiveness are both necessary in this component. We will introduce a greedy method which runs fast and returns meaningful results. After the concept nodes are organized in connected components, we further adopt the personalization step to rank

these generated components so that interesting tweets will be captured by the user conveniently. Finally, the top k ranked subtrees will be returned to the user. In the following sections, we will explicitly explain the principle of these important components.

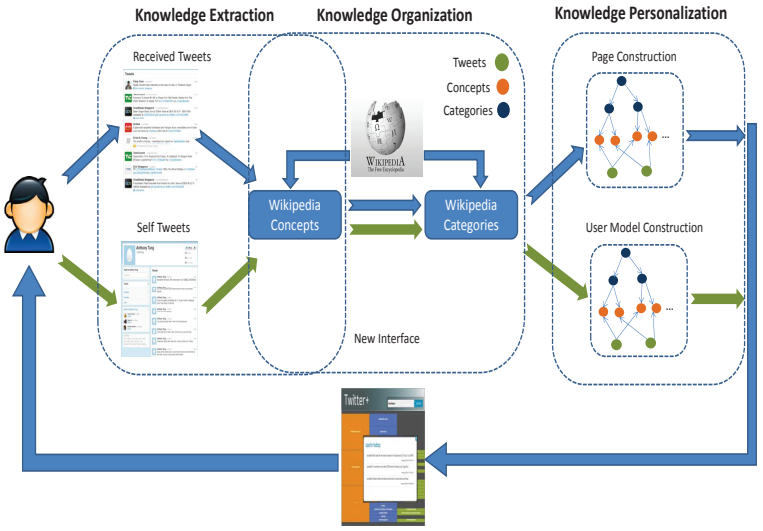


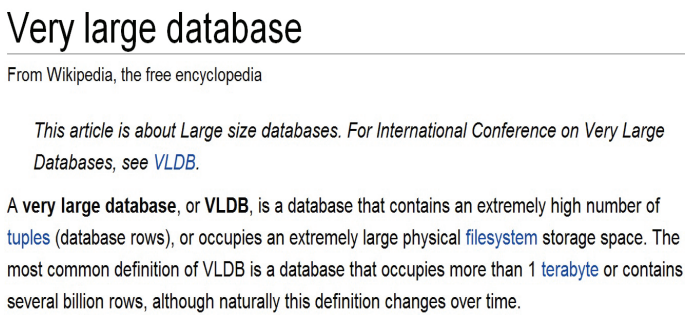
Figure 4.3: System Framework

4.2 Mapping Tweets to Wikipedia

There have been a bundle of works dealing with knowledge extraction [62, 31]. In this chapter, we utilize Wikipedia as our knowledge base and our purpose is to build a one-to-many mapping function between each tweet to the concept nodes in the knowledge graph. Our extraction solution consists of two steps: keyphrase extraction and concept identification.

4.2.1 Keyphrase Extraction

Since a tweet itself is short and noisy, it is a challenging job to extract keyphrases. We utilize Wikipedia to evaluate the importance of each phrase. In other words, given a Wikipedia concept as well as its related article, we need to identify which phrases are important. Figure 4.4 shows such an example. In this article, some terms are highlighted in blue color. We consider them more powerful than others in understanding the topic. Depending on this idea, we pre-process the tweets with the purpose of removing unnecessary words. Note that each highlighted term is hyperlinked to its corresponding concept.


Very large database

From Wikipedia, the free encyclopedia

This article is about Large size databases. For International Conference on Very Large Databases, see VLDB.

A **very large database**, or **VLDB**, is a database that contains an extremely high number of tuples (database rows), or occupies an extremely large physical filesystem storage space. The most common definition of VLDB is a database that occupies more than 1 terabyte or contains several billion rows, although naturally this definition changes over time.

Figure 4.4: An Example Wikipedia Article

This linkage information is our major source to build connections between ordinary terms and concepts.

We parse the whole Wikipedia with the wikipediaminer tool [2] to store the connections. Then we extract all n-grams ($n = 3$ in our implementation) from each tweet. Given a gram, we first check whether it appears in the database. If not, we simply drop it since it is never highlighted by any user. Otherwise, we adopt the *Keyphraseness* measure [63] to evaluate its importance. The philosophy is that the more times a word highlighted, the higher probability it is deemed as critical. The *Keyphraseness* score of a gram w is computed by dividing the total number of articles where the word appears, denoted by $T(w)$, by the number of articles where the gram is highlighted, denoted by $H(w)$. The equation is

$$K(w) = \frac{H(w)}{T(w) + \lambda}. \tag{4.1}$$

λ is a parameter to guarantee that we will not mistakenly value a word highly if it appears seldom in Wikipedia. We only keep the gram whose keyphraseness score is above a certain threshold.

4.2.2 Concept Identification

After extracting the keyphrases from a tweet, the next step is to map them to the related concepts. There have been several works handling this task [62, 56]. Since this is not the main focus of our chapter, we simply adopt the method described in [56] to solve this problem. Compared to [62], the work in [56] relies more on the context to determine the concept selection. In this method, several concept candidates are first retrieved for a word, such as the concepts "Apple.Inc" and "Apple" for "apple". Then it weights the candidates

[2] http://wikipedia-miner.cms.waikato.ac.nz/

based on its context. Finally, the one with the highest score is chosen as the corresponding concept for the word. More details could be refereed in Chapter 3.

4.3 Knowledge Organization

We repeat above process on each tweet in the tweet stream. When a set of concepts is extracted from the tweet stream, how to connect and organize them is a challenging problem. our purpose here is to connect similar concepts together, so that users can quickly find the inter-connections between tweets. Many algorithms have been developed to solve this issue, but for longer documents. For example, in [43], they compute the similarities between concepts and then apply traditional hierarchical clustering methods to group them. However, we argue that there are three drawbacks of the traditional methods. First, it is time consuming to compute pair-wise similarities between concepts considering their large volume. Second, it still fails to reveal the inner connections between concepts, such as why they are connected. Third, the label generation could be a possible method to ease the above problem. However, the label generation is usually separated with the clustering process.

In this section, we propose to integrate the clustering and label generation together. We know that the concepts are essentially leaf nodes in the knowledge graph. Two concepts will be connected to the same category if they are semantic related. Then the category could be used to explain why they are connected to some extent. For example, “iPhone” and “Macbook Pro” will be connected to the same category “Apple product”.

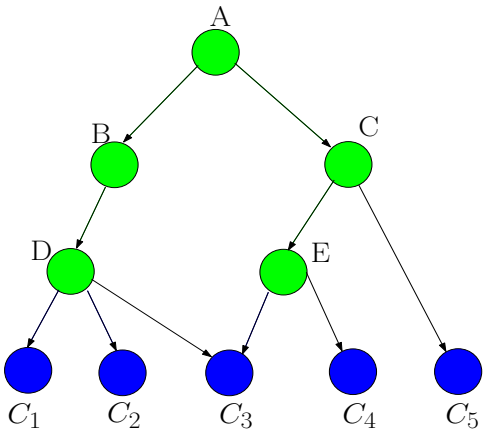


Figure 4.5: Construct subtrees covering all the concept nodes

Therefore, we propose to organize the concepts based on their shared parent categories. For example, Figure 4.5 is a subgraph obtained from the knowledge graph. The bottom

nodes $C_1, C_2, \ldots, C_5$ denote 5 concepts while the upper nodes are the categories. The concepts could be organized in different ways. For example, one possible organization is to group C_1, C_2, C_3 together since they share the same parent D. Correspondingly, Combining C_3, C_4, C_5 could be another organization. Following the same idea, 5 possible organizations are produced based on the 5 category nodes in the subgraph. The distinct points of them are that why they are connected (different parent categories) and what concepts they contain. In summary, given the concepts extracted from a tweet stream, a number of organizations could be set up according to their different parent categories. Then based on each parent category, we can obtain a subtree structure from the knowledge graph to denote the corresponding concept organization. From now on, we define each organization as a subtree and provide its formal definition as follows:

Definition 4.5. *Subtree S of tweet stream T*
Given a tweet stream T and the concepts $C(T)$ extracted from T, a subtree $S = (r, V, E)$ could be built from the knowledge graph. V represents the concept nodes C' where $C' \subseteq C(T)$ and root r of S denotes a common parent category of C'. We call the root category as the label of the subtree. The edges in E are derived from the links between r and V in the knowledge graph.

We further define the covering ability of subtree S as $Cover(S, C(T)) = C'$. Here, we do not require a subtree to cover all the concepts in $C(T)$, but to make the subtree meaningful, we only consider subtrees which cover at least two concepts. During the subtree generation process, if a category node A contains a set of concept nodes and B is a parent node of A which contains the same set of concept nodes, we consider A a better result than B as subtree A is more compact but with the same quantity knowledge. Then the subtree built based on B could be pruned, such as the subtree built on B and D in Figure 4.5. This property is essential to reduce the optimal subtree search space in the latter section.

Since the concepts could be covered with different categories, then a number of subtrees could be built. To find the best set of subtrees covering all the mapped concept nodes, we need to assign a weight to a subtree as the evaluation of its quality. Our subtree weighting function takes into account the following properties, such as concept relevance or subtree representability. Besides, several other factors will affect the tree weight as well, such as the tree height or the root category generality. Note that the smaller the weight, the better the result.

- **Concept Relevancy**. When a tweet stream is mapped to a set of concepts, some of the concepts may be more powerful than others. We assign a weight to determine

the relevance score of a concept. Since the weight of a subtree is an aggregation of concept relevance, we assign a lower weight to a subtree if it contains concepts which are more relevant to the stream.

- **Subtree Representability**. Since a concept could belong to several subtrees, we need to measure the representability of the subtree to the concept. In another word, we prefer the subtree which can better represent corresponding concepts.

In the following part, we will present our scoring function and prove that the problem can be reduced to the weight set cover problem, which is NP-Hard.

4.3.1 Subtree Weight

The weighting function for a subtree S with respect to the tweet stream T, denoted by $\mathcal{W}(S,T)$, needs to take into account the tree size and the concept weight. We first identify the weight function $\mathcal{W}(c,T)$ of a concept node c. Given a tweet in the stream, as mentioned before, we extract keyphrases and detect related concept nodes using the association between keyphrase and concept, which is built offline. Each keyphrase p has a probability $P(c|p)$ to be associated with c. We utilize the concept frequency CF to measure the importance of a concept. If a concept is mapped with multiple tweets, we consider it more important. Finally, we can get

$$\mathcal{W}(c,T) = \sum_{p \in T} P(c|p) * CF(c).$$

To measure the relevance $\mathcal{R}(c,S)$ between concepts and its covering subtree, we identify the relevancy in two aspects. First, we consider similarities among the concepts covered by the tree. If the concepts are semantically close to each other, we consider the tree as coherent. Second, the tree height should be considered as well. According to the hierarchy of the knowledge graph, the larger distance between the tree root to its covering concept, the less representative it is. Formally, we define $\mathcal{R}(c,S)$ as follows:

$$\mathcal{R}(c,S) = \frac{\sum_{c_i \in S.V \bigwedge c \neq c_i} Sim(c,c_i)}{h(S) * |S.V|}$$

We employ the method in [64] to compute $Sim(c,c_i)$. The higher value means the two concepts are more relevant. $h(S)$ is the tree height which is the longest path in the tree. $|S|$ represents the number of concept nodes contained in S.

Another concern is that a more general category will naturally connect to more concepts in the knowledge graph. For example, in the extreme case, the root of the knowledge graph will connect to all the concepts in Wikipedia. However, the tree formed based on the root category is useless to the users. Therefore, we need to normalize the weighting function to reduce the side effect. Given a subtree S, the normalization factor $\sigma(S)$ is defined as:

$$\sigma(S) = \frac{d(S)}{l(S)}$$

where $d(S)$ represents the depth of the root node of the subtree in the whole knowledge graph. A small value of $d(S)$ means the subtree is close to the root node of the knowledge graph, i.e., the category is more general. $l(S)$ is the outdegree of the root node in the knowledge graph, a generative category will be of a larger outdegree, such as "United States of America". In summary, the final weighting function $\mathcal{W}(S,T)$ for a subtree S can be defined as:

$$\mathcal{W}(S,T) = \frac{1}{\sigma(S) \cdot \sum_{c \in S.V} \mathcal{W}(c,T) \cdot \mathcal{R}(c,S)}$$

4.3.2 Optimal Subtree Selection

Given the weighting function of a subtree, our goal is to pick m subtrees so that they cover all the required concept nodes and the total weighting is smallest. We formalize the problem as follows:

Definition 4.6. *Optimal Subtree Selection*
Given a set of concepts C and their covering subtrees $\mathtt{S}$, find $\mathtt{S}' \subseteq \mathtt{S}$, so that $Cover(\mathtt{S}', C) = C$, and the weight $\sum_{s \in \mathtt{S}'} W(s)$ is minimized.

As introduced before, in Figure 4.5, 5 subtrees rooted at A, B, C, D, E could be built. Our goal is to find a set of subtrees from them which cover all the concept nodes and are assumed as the best organization solution. In this case, $\{D, E\}$ is not a candidate because the set does not cover C_1 to C_5.

We prove that the above problem is NP-hard.

Propositon 4.1. *The Optimal Subtree Selection problem is NP-hard.*

Proof. This proposition could be easily proved by a reduction from the weighted set cover (WSC) problem [47]. We reduce an instance of WSC problem to an instance of the Subtree selection problem as follows: for every element e in the WSC problem, we create a concept node c. For every subset g of the WSC instance, we create a subtree S which covers the corresponding concepts nodes. The weight of S equals to that of g. Given this mapping, it is easy to show that there exists a solution to the WSC problem which minimizes the weights if and only if there exists a solution to the subtree selection problem. □

4.3.3 Subtree Selection Solution

In this section, we adapt a greedy algorithm [20] to solve the optimal subtree selection problem. This algorithm will not only select the subtree set but also order them.

Algorithm 4.1: Greedy Algorithm for Subtree Selection Problem

Input: tweet stream T, Extracted Concept $C(T)$, Subtree Set $\mathtt{S}$
Output: A subset of $\mathtt{S}$ which minimize the weight function

```
1 begin
2     U = C(P), R = ∅;
3     while U! = ∅ do
4         let S_i be the subtree that has the minimum weight w_(S_i)/|S_i ∩ U|
5         R = R ∪ S_i;
6         U = U\S_i;
7     return R;
8 end
```

We illustrate the algorithm in Algorithm 4.1. Given all the possible subtrees, we first order them by the score function.

$$Score(S) = \frac{W(S,T)}{S \cap U} \tag{4.2}$$

U is the set of elements that are not covered as yet. According to the function, a subtree with a higher weight and covering more concepts will be selected out first. Both of the two criteria make sense when we rank and present the subtrees to users. Another implicit advantage of the formula is that it considers the redundancy between subtrees, as well. This property expresses itself in the formula $S \cap U$. It suggests that at each step it favors the subtree which contains more new concepts compared to those already being covered.

We use a result container to keep the final solution subtrees. At first, the container is empty. We evaluate all the possible subtrees according to equation 4.2 and pick the one

which minimizes the equation. We repeat the whole process until all the concepts are covered with at least a subtree. Then we return the subtrees in the result container as the solution.

It is easy to see that the complexity of the algorithm is $O(C(T) * log(|\mathtt{S}|))$ as the outside loop operates for $O(C(T))$ time and the subtree which satisfies equation 4.2 could be found in $O(log(|\mathtt{S}|))$ time. Furthermore, according to [47], this greedy algorithm achieves a $H(d^*)$ approximation to the optimal results where d^* equals to $|S.V|$ where S is the tree which covers most concepts.

4.4 Knowledge Personalization

Given a set of subtrees returned by the greedy algorithm, a naive solution is to display them in the rank of their selection order. However, such an arrangement does not consider any influence from the targeting user. In this section, we consider how to re-order the subtrees when personalization is involved.

A subtree will be assigned a higher score if it is matching more with the personal interests. In this chapter, we use the outgoing tweet stream of a user as the source to identify his personal interests. We adopt the technique presented above, including knowledge extraction and knowledge organization, to transform the outgoing tweet stream into subtrees. The problem becomes identifying the relevance score between a subtree and another set of personal subtrees. The best subtree is considered as the most relevant one to personal interests.

There exist several different ways to compute the similarity between tree-structure items. For example, we can decompose subtrees into a set of nodes and utilize the Jaccard's distance to measure the difference. This method is efficient but sacrifices the quality. First, it only considers the exact match of nodes but ignores the textual similarity between them. Second, it fails to measure the structural connections due to the decomposition. Another way is to adopt the tree edit distance [9]. This metric is defined as the minimal number of edit operations to transform one tree to another. However, computing tree edit distance suffers an expensive computational cost. To avoid above problems, we propose a tree kernel method. Compared to above methods, our kernel based approach can be computed in linear time and capture both the structural and textual similarity.

Tree kernels have been originally designed based on the idea of efficiently counting the number of tree fragments that are common to both argument trees. A larger function value means a more similar result. Different fragments of trees could be defined according to various situations, such as subtrees or subset trees [91]. In our case, the tree fragments are path based. Each fragment is formally defined as a path which starts from the root and ends at any internal node.

$$F(S) = \forall_{n \in S} Path(S.root \leftarrow n) \tag{4.3}$$

There are three reasons that encourage us to process in this way:

1. Since our tree is generated from Wikipedia. Each path is enriched with semantic meanings. Two trees are similar if they share a common path. Therefore, there is no need to require a matching between subtrees as in subset tree splitting.

2. The path based fragments provide more chances to find a match between trees. It is equivalent as achieving "Query Expansion" effects in finding similarities between a user and a subtree.

3. Intuitively, the match between longer paths is more valuable than the shorter ones in finding similarities. This idea could be naturally expressed in our method as more fragments can be derived from the longer path.

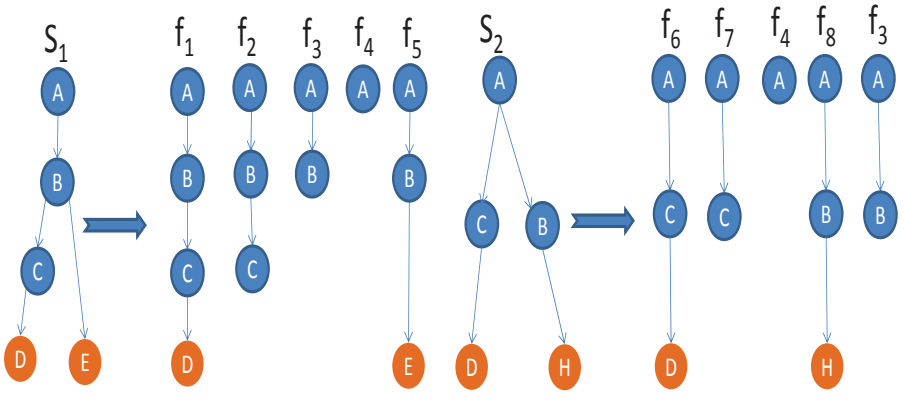


Figure 4.6: Example of Fragment Generation

We give an example to illustrate fragments generated for two trees in Figure 4.6. In the original tree kernel, it takes each fragment equally and only considers the structures. For example, the kernel score $\kappa(S_1, S_2) = 2 * 1 + 2 * 2 = 6$ only because they share the substructures f_3 and f_4. In our case, we hope to take the node contents of the fragment

into consideration when comparing them. Since nodes in fragments contain category or concept information, we could evaluate the content importance of two fragments from two subtrees that have the same structure. Let f_a be a fragment in S_a and f_b for S_b, we represent each of them as a vector $v = (w_1, w_2, ..., w_t)$ with each dimension corresponding to a node. Each element in v is evaluated by the metric introduced in Section **??**. Then for two fragments, we obtain:

$$\kappa_f(f_a, f_b) = v_a \cdot v_b \tag{4.4}$$

where v_a and v_b are weighted vectors for f_a and f_b separately. $v_a \cdot v_b$ is the dot product.

Consequently, the tree kernel between two subtrees S_a and S_b is defined as follows:

$$\kappa(S_a, S_b) = \sum_{n_a \in S_a} \sum_{n_b \in S_b} \Delta(n_a, n_b) \tag{4.5}$$

whereby

$$\Delta(n_a, n_b) = \sum_{i=1}^{|F(S_a) \bigcup F(S_b)|} I_i(n_a) I_i(n_b) \kappa_f(f_i^a, f_i^b)$$

and where $I_i(n)$ is an indicator function which determines whether fragment f_i is rooted in node n.

Given the tree kernel, the relevancy between a subtree S and user u is defined as:

$$Rel(S, u) = \kappa(S, \mathbb{S}(u)) \tag{4.6}$$

where $\mathbb{S}(u)$ is the subtree set generated from the user's outgoing tweets. Finally, the subtrees generated from last sections will be re-ordered based on equation 4.6 and returned to the user.

4.4.1 Efficient Tree Kernel Computation

Computing tree kernels ultimately consists in comparing all the fragments of two trees S_a and S_b. It is concluded that the worst case time computational complexity of the kernel is $O(|S_a| * |S_b|)$. The quadratic cost may become prohibitive when the number of subtrees becomes large.

We propose to leverage the suffix tree model [106] to manage and facilitate the subtree kernel computation. A suffix tree is a data structure that is originally used to present the

suffixes of a given string. The edges of the suffix tree are labeled with strings, such that each suffix of the given string corresponds to exactly one path from the tree's root to a leaf.

After a closer observation on suffix trees, we realize that our subtree fragment generation could be seamlessly integrated into its building process. Back to our method, for each path of S, we are able to construct a string where the character is the label of nodes that along the path. Note that the characters will be arranged by the reverse order of the labels. Apparently, each fragment we defined before represents a suffix of such a string. For example, the reverse label of fragment f_2 (CBA) in Figure 4.6 is a suffix of the string constructed by the reverse order of path $A \rightarrow D$ $(DCBA)$. All the suffixes of strings built from above paths constitute a subtree's fragments. Figure 4.7 is an example built for trees in Figure 4.6. Each path from the root to a leaf corresponds to a fragment. Each suffix leaf node has one box attached to it designating its originated source. The first number in each box designates the subtree/user of origin, the second number designates the fragment vector. If a fragment origins from multiple sources, then the box will contain multiple series numbers, such as BA. Suffix tree could be constructed efficiently by implementing the

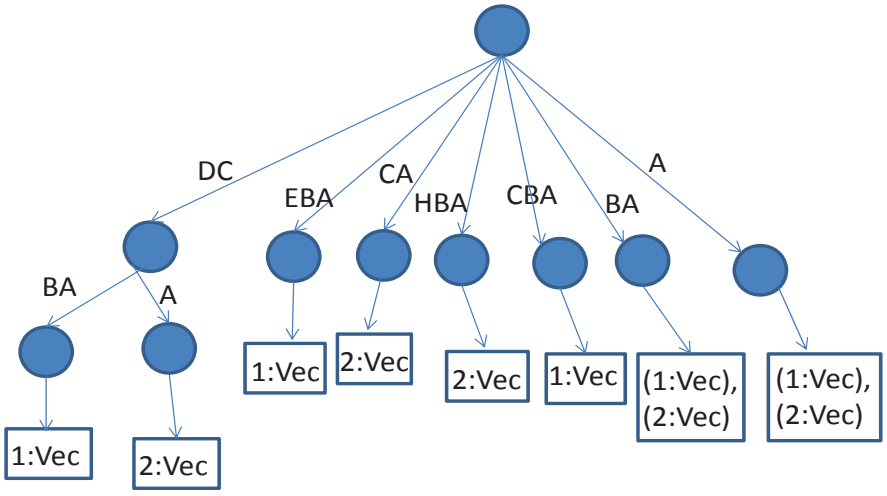


Figure 4.7: Suffix Tree of Two Subtrees

Ukkonen' algorithm [89]. By exploiting a number of algorithmic techniques, Ukkonen's algorithm reduces the construction time of suffix tree to $O(n)$.

In summary, before computing the kernel similarity of a subtree to a user, first all the subtrees are split into individual paths, then a suffix tree is built based on the paths. The box of each node will contain at most two numbers: the subtree id or the user id. When computing the kernel, we only need to traverse the leaf nodes of the suffix tree, if there

is a box which contains more than one number, we would know that this fragment could contribute to the kernel computation. Therefore, the efficiency of kernel computation could be improved. In addition, the suffix tree is also utilized as the index structure of subtrees which facilitates their future usage.

4.5 Experimental Study

In this section, we conduct a comprehensive user study to show the effectiveness of our methodology. In particular, we test the effectiveness of knowledge organization, personalization and visualization via Amazon's Mechanical Turk.

4.5.1 Experimental Setup

In the following, we briefly introduce the experiment setup in Amazon's Mechanical Turk and Wikipedia knowledge base.

To make the experiment results fair enough, we hire workers from Amazon's Mechanical Turk [3] to evaluate our performance. Amazon Turk offers an access to a community of human subjects. Acting as a tool, it distributes small tasks that require human intelligence. To guarantee the quality, we perform a qualification test on the workers before they join our evaluation. The qualification requires a worker must be active in Twitter, such as publishing more than 50 tweets and having at least 10 followings. Besides, their tweets should be written in English since our method relies on the Wikipedia archive in English. Only those who have passed the qualification examination are allowed to participate our evaluation procedure. The qualification is implemented by Turk API. For each participating worker, we collect both of the outgoing and incoming tweet stream. The first part is used to build a personal interest model and the second is used for knowledge organization and visualization. Some statistics of the workers are listed in Table 4.1. Each worker will only be responsible for the results produced from his own data. From the table, we can see that each worker has a plenty of incoming and outgoing tweets so that we are able to obtain accurate feedback for the evaluation.

[3] https://www.mturk.com/

Table 4.1: Statistics of Workers

Number of workers	50
Avg. number of followings	283
Avg. number of incoming tweets	572
Avg. number of outgoing tweets	2451

Wikipedia

The Wikipedia data used in this chapter is obtained from University of Waikato [4]. It was released in 22nd, July, 2011 and processed by the Wikipedia Miner tool. The data set contains $739,980$ categories and $3,573,789$ concepts in total. Besides, there are $80,381,903$ links in the knowledge graph.

4.5.2 Effectiveness of Knowledge Organization

Table 4.2: Comparison of Different Methods

Method	**MAP**	**Mean Prec@10**	**MRR**
$TFIDF$	0.569	0.351	0.609
$Yahoo!$	0.5723	0.382	0.597
$Subtree\ Label$	0.761	0.558	0.801

For each worker, we present him the labels of the top 10 subtrees generated by our algorithm. Each worker is asked to judge whether the labels are "relevant" or not based on their data. For comparison, the workers will be presented with two other lists which include the top 10 keywords generated by $TFIDF$ weighting function and $Yahoo!API$ [5]. The evaluation metrics include *Mean Average Precision (MAP)*, *Mean reciprocal rank (MRR)*, and *Precision@n*. *MAP* rewards approaches returning relevant results earlier, and also emphases the rank in returned lists. *MRR* gives us an idea of how far we must look down in the ranked list to find a relevant result. *Precision@n* is the fraction of the top-n results that are relevant.

We give an overview of the comparison in Table 4.2. As observed from the table, our method is significantly better than the other two in all the three metrics. Compared to the keyword labels, our method captures a more general and precise topic. For example,

[4] http://wikipedia-miner.cms.waikato.ac.nz/
[5] http://developer.yahoo.com/search/content/V1/termExtraction.html

"bouquet", "bridal" and "ceremony" appear in a worker's outgoing tweets. We can successfully build a subtree, which includes those tweets and derive "Category:Wedding" as the subtree label, though "wedding" does not appear in any of the tweets.

To illustrate our results better, we present the top 10 subtree labels which are generated from 4 celebrities' tweets in Table 4.3. The labels denote the topics of their tweets. These 4 people are from different areas so that we could demonstrate the generality of our method. As we can see from the table, occasionally, inappropriate labels may be produced, such as "Biology" for "WayneRooney". We analyse the reason as that many of the tweets published by the user are meaningless. Then concepts from those tweets would be connected to a very general category and form a higher weighted subtree since it covers many concepts, such as "Member states of the United Nations". However, the representability of such subtrees are weak or sometimes inaccurate. We plan to investigate how to handle this problem in the near future. However, as shown in the Table, the majority of the results are quite reasonable. This demonstrates that our method is capable of categorizing tweets into meaningful subtrees. For example, in Bill Gates's tweets, we successfully capture that currently he cares more about "philanthropy", and sometimes mentions about "economics" and "technology". This also illustrates that users' interests can be reflected from their tweets.

Table 4.3: Example Labels for Celebrities

BillGates	BarackObama	Oprah	WayneRooney
Member states of the United Nations	Presidents of the United States	Oprah Winfrey	Premier League players
Global health	Member states of the United Nations	Member states of the United Nations	Premier League clubs
Economics	Barack Obama	Cities in Texas terminology	Association football
Infectious diseases	States of the United States	English-language television series	Biology
Public health	Taxation	Grammy Award winners	Member states of the United Nations
Nobel Peace Prize laureates	Economics	English-language films	Ethology
American philanthropists	Elections	Christian terms	Grammy Award winners
Education	Political terms	American film actors	Interdisciplinary fields
Development	Debt	2000s American television series	Association football in Europe
Technology	Educational stages	Television terminology	Ball games

4.5.3 Effectiveness of Kernel Similarity

In this part, we intend to evaluate the effectiveness of our proposed kernel similarity function. Although we are lack of ground truth here, the philosophy of our evaluation could be described as follows:

Compared with a random selected user who is far away from a worker in the social network, the tweets of the worker are intuitively more similar to those from the people he is following since the following relation in Twitter is normally built on common interests [71].

Therefore, based on a user's tweets, an effective similarity measure should be able to rank his followings higher than those randomly chosen users. For each worker, we compute the similarity between his tweets and each of his followings'. These are used as the positive test examples, and their relevancy scores are set to be 5. As the negative test examples, we select the equivalent number users that the worker does not follow. We process them by the same procedure. Their relevancy is set as 1. Then different similarity functions are compared by the extent to which they rank the positive test users over the negative users. Specifically, we use Normalized Discounted Cumulative Gain $(NDCG)$ [44] as the measurement metric. This metric takes both relevancy and ranking order into consideration and is widely used to measure the result quality. Three alternative methods are compared with our method:

1. $Bag\ of\ Words(BOW)$, models a user in a $TFIDF$ weighted vector by his tweets and utilizes cosine function to compute similarities between users.
2. $Bag\ of\ Concepts\ and\ Categories(BOCC)$, similar to BOW except that the elements in the vector are concepts and their connected categories as in the subtrees.
3. $Jaccard\ Similarity$, builds the same model as BOW but uses Jaccard similarity as the measurement.
4. $Subtree\ Kernel$, models a user by a set of subtrees and computes the similarity by our kernel method.

Table 4.4 shows the comparison results across the four methods. Two conclusions are drawn from the results. First, Wikipedia semantics can improve the similarity computation between short text, such as tweets, since both of the two bottom methods outperform

Table 4.4: Comparison of Similarity Functions

Method	**NDCG Score**
$Jaccard$	0.408
BOW	0.423
$BOCC$	0.470
$Subtree\ Kernel$	0.554

the upper two. We believe the reason lying in that $BOCC$ representation reduces the ambiguity problem in tweets. This problem is more serious in short text than the normal documents as there is less context information to help understand the topics of tweets. Second, structural information in Wikipedia is valuable to improve the performance further. Compared to the third method, our kernel method achieves a 17.8% increase of the performance.

4.5.4 Effectiveness of Knowledge Personalization

The personalization is essentially a re-ranking step to emphasize those which are more relevant to a user's personal interests. To evaluate whether our method yields a gain, we need to transform our subtree results into a plain list of tweets since in current Twitter or many other interfaces, the tweets are shown in a list. To make them comparable, we leverage the evaluation strategy in search result clustering area. According to the method in [14], our list is produced by choosing the tweets from the top ranked subtree and going down until we reach a pre-set number of the total unique tweets. Such a subtree linearization would preserve orders in which subtrees are presented.

In this experiment, we compare our personalization method with the other three tweet lists. One is simply ordered by the post time of the tweets. The other two are ordered by cosine similarities between incoming and outgoing tweet streams or the streams mapped to Wikipedia. We did not compare with the Jaccard similarity measure as it shows worse performance than the cosine similarity in the above experiment.

1. $Timeline$, orders the incoming tweets by their received time. This is provided by existing Twitter system.

2. BOW, models tweets in `tf-idf` weighted vectors and orders them by the cosine similarity with users' outgoing tweet stream.

3. $BOCC$, replaces the elements in above vectors to concepts and categories mapped by tweets.

4. $Subtree$, ranks the tweets based on our proposed methodologies.

In the experiment setup of Amazon Turk, we only consider top 30 tweets in each ranked list. Each worker is asked to evaluate the relevancy of the ordered tweets according to his interests. To evaluate the rank lists, we employ the $NDCG$ score as in previous experiments.

Table 4.5: Comparison of Different Lists

Method	**NDCG Score**
$Timeline$	0.303
BOW	0.324
$BOCC$	0.346
$Subtree$	0.383

Table 4.5 compares the average NDCG scores of the ranked lists generated by different methods. As shown in Table 4.5, the score for "Timeline" is relatively low. It indicates that current ordering of tweet stream can not satisfy the users. However, our *Subtree* method achieves the best performance and shows a significant improvement.

4.5.5 User Survey On Visualization

Table 4.6: User Survey

Question	Positive	Negative
Which interface do you like?	38	12
Will you use the interface in the future?	35	15
Is it useful for browsing and exploring tweets?	40	10
Are you satisfied with the generated subtrees?	33	17

Besides the above result evaluation, we also conduct a survey to collect user feedbacks on our new interface. The survey is designed with several questions listed in Table 4.6. Their responses are shown in the last 2 columns. The "positive" column represents our interface while "negative" for current Twitter interface. As shown in the table, most workers express intensive interests on the knowledge based presentation, feel conveniently to view

the tweets and are satisfied with the generated subtrees. The main user complaint is that sometimes there are too many subtrees generated. Users expect more compact representations. For this issue, we plan to let users adjust parameters which can control the size of subtrees. In addition, some workers also suggest that they expect to see such a knowledge based tweet presentation for a particular group instead of the whole followings. As our current implementation is mainly content-based, we plan to investigate more on the user interactions in the future.

4.6 Discussion

The knowledge view of tweet stream generated by our method is certainly not perfect and could be improved on many aspects. The major purpose of this work is to provide users more options to browse the information online. The pitfalls and drawbacks of our system pose several open challenges which could possibly form new research directions. We plan to prompt our investigations along the following lines:

1. Improve the concept mapping accuracy. Since our subtrees are trying to categorize similar tweets together; therefore understanding the tweet intents becomes quite significant. The accuracy of concept mapping hinders us from further improving the quality of the generated subtrees. Although there are many methods, which are supposed to fulfill this task [62], their ability of handling the large scale Twitter data is still in doubt. Currently, we map the tweets to their corresponding Wikipedia concepts mainly by cleaning up the data and then relying on its context. We plan to investigate more NLP techniques and batch processing methods to address this problem.

2. Optimize subtree structures. Currently, the backbone of each subtree is fully relied on the links between categories in Wikipedia. In many cases, the links suggest a meaningful hierarchy. However, we should note that many of them are still redundant and noisy. For example, the shortest path between "iPhone" and "Category:Apple Inc." is 4 although their relation seems quite close from the common sense. This problem will reduce the expressivity of our subtrees and increase the difficulty for users to access the information. We plan to clean up the hierarchy of Wikipedia to make it more succinct.

3. Combine context information into subtree orderings. For the time being, we primarily order the tweets by their contents. However, the ranking of a tweet is not

only decided by its topics but also many other factors, such as where it comes from or is it private or broadcasted news. The context information plays an essential role in determining the usefulness of a tweet. Next, we plan to combine our work with those context information to improve the presentation of current methodology.

4.7 Summary

In this chapter, we present a new tweet organization method which provides users an alternative interface to browse their tweet stream. The interface is designed to ease the information overload problem. By fully utilizing the Wikipedia contents with their well-organized hierarchy and high-quality links, we build a knowledge graph and derive a number of subtrees to categorize the tweet stream automatically . To facilitate the user's reading further, the subtrees are ordered based on their relevancy to the user's preference and labeled by explicit terms. In summary, the contributions of the chapter include:

1. We add a knowledge dimension to the tweet stream so that users can find useful or interesting tweets conveniently.
2. We propose a hierarchical manner to organize tweets according to structural information in Wikipedia. In addition, we devise a kernel method to measure similarities between tweets.
3. We implement a complementary interface to view Twitter.

Finally, we conduct a comprehensive user study on our methodologies, including the subtree label quality, kernel method effectiveness, subtree relevancy and etc. The feedbacks from users are promising. Based on the empirical results, we can draw the conclusion that the alternative interface is effective to enrich the user's exploring experience in Twitter.

CHAPTER 5

PERSONALIZED USER TAG PREDICTION IN SOCIAL NETWORK

Modeling and predicting user interests are urgently needed in many systems and services to meet individual user's needs [87]. For example, in Amazon[1], they expect to capture user' interests precisely so that the recommendations of related products can be done even without user notice. Towards this goal, existing works have modeled user interests using different sources of profiles such as the explicit demographic or interest profiles, or implicit profiles based on query logs, search result clicks and etc [99]. Recently, with the proliferation of online social networks, e.g., Facebook, Twitter or Delicious, more valuable sources are available to assist in handling this task since these services record various ego-oriented data such as their status, comments, activities and social connections. Compared to previous sources, e.g., query logs, the new type of data reflects users' interest more accurately and thus has prompted many more personalized services, such as the social network advertising. Because of its great potential and importance, more and more researchers have concentrated on predicting user interests in order to provide personalized results.

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

A typical interest prediction problem is the tag prediction in social networks. There are two reasons that researchers are inclined to use tags as demonstrations for interest prediction. First, since tags are always used as metadata to describe and annotate web resources, they are generally assumed as an explicit indication of user interests [33, 75]. Second, the tag data is easy to obtain and the results can be evaluated in a relatively straightforward way.

Traditionally, the tag prediction problem is referred as *resource tag prediction*, which predicts related tags to a particular object. There are two main directions for handling resource tag prediction problem. One is the content-based approach and the other is the graph-based approach. Content-based methods [54] model tags from textual information, including resource contents or user history. Graph-based approaches [103, 29] usually form a graph by the connections between users, resources and tags. Then they apply unsupervised or supervised approaches to learn the affinity between the user and tags. In these works, tags are mainly predicted for resources. However, in this chapter, we propose a new definition of the tag prediction, the *user tag prediction*, in which the prediction targets at the user instead of the resource. To be more concrete, given a tag, we expect to predict whether the user will use it or not in the future. Then related information of the tag will be pushed to users in time. The information could be contents contributed by other users or the advertisements from enterprises. Since the prediction is dynamic, the recommendation will be adjusted from time to time according to their evolving interests. In this way, we believe that the users will be satisfied better.

In this chapter, we describe a systematic study of numerous sources for user tag prediction in social networks. We provide an in-depth analysis of what, why and how these sources play roles in deciding whether a tag will be used by a user. Based on the analysis, our proposed framework learns the tag prediction model for each user using four types of information namely, (i) *Frequency:* a tag's occurrence infomation in the user's profile if there is any, (ii) *Temporal:* temporal usage or patterns of a tag, (iii) *Correlation:* the correlations of a tag with its surrounding tags, and (iv) *Social Influence:* the social influence on users' tag usage. We have analytically studied the utility of each source based on its effectiveness for prediction. Finally, we extract a number of features according to the analysis and model the prediction as a classification problem.

The rest of the chapter is organized as follows. In Section 5.1, we present the statement of our problem. Section 5.2 describes the data used in this chapter and our analysis based on the real data set. In Section 5.3, we illustrate the evaluation methodologies and results of our prediction tasks. Finally, we make the conclusion in Section 5.4.

5.1 Problem Overview

Social tags have recently emerged as a popular way to allow users to add metadata in the form of descriptive terms to describe web resources. Since added by the user, the tags show the potential to model user's interests [85]. For example, a user who is interested at the topic of "Big Data" may browse related web pages and use "big data" as the corresponding tags. In this case, the tag will reflect the user's interests to some extent. From this point of view, predicting user's future tags will help understand user's interests and hence benefit many applications. However, these assumptions require us to predict tags for users instead of resources. Predicting user's future tags accurately is a challenging problem since user interests drift over time. Briefly, user can keep using his past tags and/or adopt a new tag. In the first case, user may use some of his current tags or reuse some of his past tags again. In the later case, the newly adopted tags may be influenced by his social neighbors or evolved from his past interests or may be an external popular interest. In summary, there are several reasons which can explain why a user would accept or abandon a tag. Since there is no targeting resources, we derive the knowledge from the users' historical data. Then it will be referred as the user profile.

Definition 5.1. User Profile:
Given a user u, T be the set of tags t and $\mathcal{T}$ be the set of timestamps τ, S is the set of all records s, representing the relations among the three types of objects, $S \subseteq u \times T \times \mathcal{T}$. Each record $(u, t, \tau) \in S$ means that user u has used the tag t at time τ. Additionally, $\Rightarrow$ denotes the following relations in the social network. Then F_u will be the users u is following, which is $F_u = \{u' | u \Rightarrow u'\}$.

In this chapter, we propose four primary explanations which are derived from the user profile for the above phenomenon, including *frequency*, *temporal*, *correlation* and *social influence*.

1. *Frequency*. The past history of a tag is critical in determining its future. Intuitively, if a tag has been used many times in the past, it suggests that the user is always interested at the topic and therefore highly probably use the tag again.

2. *Temporal*. The temporal usage of a tag is another important source. The preference of a tag will not remain constant all the time. In certain works[3], they model the

temporal information as decaying in an exponential rate. However, we believe more accurate model can be built according to the tag evolving information.

3. *Correlation*. Correlation is the tendency of occurrences among a number of tags. The occurrence of a tag perhaps correlates with others. Co-occurrence is the most straightforward correlation. How to find correlated tags especially when there are few occurrences of a tag is the main focus in this part.

4. *Social Influence*. Social networks have provided platforms where users can see their neighbors' (friends) used tags. Many works have investigated the social influence among users. They assume social influence has a positive effect on the tag usage of a user.

According to the four types of information, we formalize the user tag prediction problem as follows.

Definition 5.2. *User Tag Prediction*
The aim of the prediction is to learn a function $\mathcal{F}$ so that given a user u and a tag t, we expect to predict whether u will use t in the time τ or not. Formally,
Input $X = \{u, t, \tau\}$; Output $Y \in \{0, 1\}$
Goal: construct a prediction function $\mathcal{F} : X \rightarrow Y$

To the best of our knowledge, we are not aware of any work that has considered all features extracted using above four sources. In Figure 5.1, we show an overview of the major steps involved in inferring user's future tags. Given a record database S, in the first step we extract various features to describe the user's tags. Once these features are discovered, we build a classifier that combines all extracted features to predict the user's future tags. In section 5.2, we describe the Delicious Dataset that we have used to analyze various features introduced in this chapter. Then, details of each feature discovery are given in the following sections. Note that we mainly focus on explaining the correlations between the features and the prediction.

5.2 Feature Discovery

First, we introduce the dataset that is utilized for explaining various concepts in this section. The dataset is crawled from Delicious and used as one of the benchmarks in HetRec

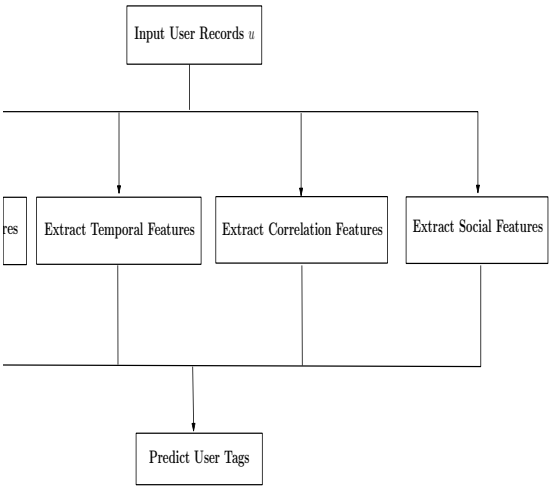


Figure 5.1: The Overall Framework for User Tag Prediction

2011[2]. The dataset contains social networking, bookmarking, and tagging information for a set of 1867 users. The tuples for user data are represented as $< user, tag, resource, timestamp >$. For each user, we also have his social connections. We build the profile for each user from the corresponding tuples and include the user relations as well. We present the statistic details of the dataset in Table 5.1.

Table 5.1: Statistics of the Delicious Dataset

Features	**Value**
Users	1867
Unique URLs	69,226
Total URLs	104,799
User Relations	7,668
Unique Tags	53,388
Total Tags	437,593
Average History Length	130 days

5.2.1 Frequency Analysis

In this subsection, we study whether the future user tags can be predicted by their past usage. Intuitively, if a tag is used many times by a user in the history, it has a higher probability to be used again. Such kind of user tags are assumed to be all time favorite and are referred as long-term interested tags. For example, a sports fan may browse sports related web pages quite often and use similar tags repeatedly. Comparatively, the tag of

[2]http://ir.ii.uam.es/hetrec2011/datasets/

low frequency could be viewed as short-term interests.

Question 1. Whether frequency has the prediction power and is the intuition valid?

We calculate the tag frequency $f(u,t)$ of tag t as the *total occurrence times* of t in the user profile of user u and given as

$$f(u,t) = |\{s|s \in S \ \wedge \ (s.t = t)\}|.$$

From the computation of the previous dataset, on average, each tag is used 1.89 times for each user and most of tags have a frequency which is less than 13 (more than 99%) within an average time period of 140 days. To obtain a clearer depiction of tags, we separate them based on their frequency. The same philosophy applies in many other places through this chapter. We display their usage in Figure 5.2.

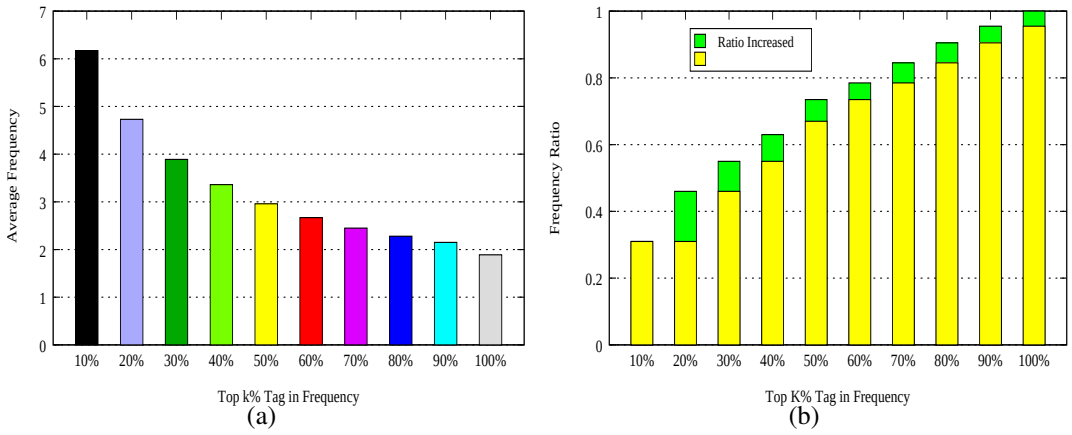


Figure 5.2: Distribution of Avg. Frequency & Frequency Ratio

In both of the two figures, X axis represents different portions of tags. For example, 10% denotes the top 10% tags which are ordered by their frequency. The same meaning stands for 20%, 30% and etc. In Figure 5.2(a), Y axis represents the average frequency for different portions of tags. As shown in the figure, the average frequency is quite different for different parts of tags. The top ranked tags have a much larger average frequency (6.17) compared to the others. Additionally, we find that most of the tags (70%) occur less than 2 times. This demonstrates that **users only use a small portion of tags of the total tag collection**.

In the next step, we present Figure 5.2(b) where Y axis denotes the ratio between the sum

of total frequency of top k% tags and the sum of frequency of all tags.

$$frequency\ ratio = \sum_{u \in U} \frac{\sum_{t \in topk\%T_u} f(u,t)}{\sum_{t \in T_u} f(u,t)} / |U|$$

As seen from the figure, top frequent tags contribute the most significant portion of the total occurrences and the ratio decreases while the k in top $k\%$ increases. For example, the top 10% frequent tags occupy more than 30% usage of all the tags compared to top 20%.This indicates that *users are inclined to reuse the frequent tags. The phenomena becomes more significant when the tag frequency becomes larger*. Both of the two figures illustrate that various tags weight quite different for each user.

In order to investigate what kind of tags are easily dependent on its previous usage, we prepare an exposure curve which is similar to [76]. However, in our case, it is designed for each individual user instead of the whole population. We say that user u is k-exposed to tag t if user u has used t at least k times in the past. Let $T(u,k)$ be tags which were at least k exposed, and let $T(u,k+1)$ be those that were $(k+1)$ exposed. Apparently, $T(u,k+1) \subseteq T(u,k)$. We then define that the probability of using the k exposed tag in the future as:

$$ER(u,k) = \frac{T(u,k+1)}{T(u,k)}.$$

From the equation, we expect to find associations between the tag frequency and it's future usage. According to Figure 5.3, we observe a positive correlation between k and the exposure rate. In particular, with the increase of k, $ER(u,k)$ becomes larger until the frequency reaches a certain limit, which is 13 in our case. As k is directly related to tag frequency, therefore, the lines in the figure suggests that if a tag is already associated with a high frequency, then the probability of its reoccurrence will become larger. This figure also helps explain why it is reasonable to use the frequency as the tag importance.

Based on above discussion, we conclude that **the frequency indeed has the prediction power. However, the power is different according to different tag frequency. It is much stronger when the tag is associated with a high frequency. Otherwise, the power is relatively not so significant.**

5.2.2 Temporal Analysis

Although the frequency is already proven to be an effective indication of the long-term interested tags, frequency alone can not handle the work very well. For example, one

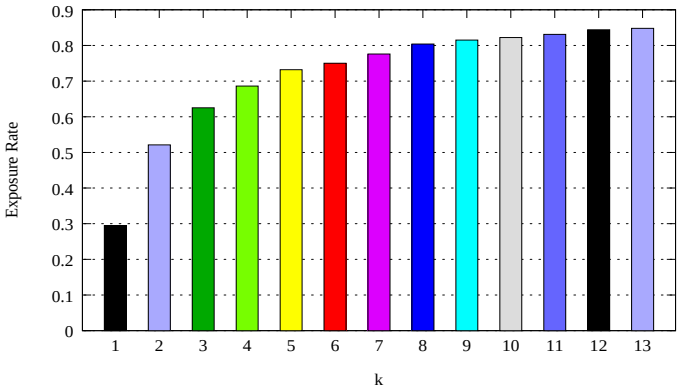


Figure 5.3: Exposure Rate for Different Frequency Tags

tag may be mentioned tens of times a long time ago but hardly referred in recent days due to the user interest shifting. Therefore, the temporal information of tags should receive enough attention. In this section, we study the temporal dynamics of user tags. In particular, we are interested to extract temporal features of tags.

First, we define the time series that will be used throughout this chapter to answer following questions.

Definition 5.3. *Time Series*
A time series for tag t of user u, denoted as $\mathcal{T}_{u,t}$, is a time-ordered sequence of data points $\mathcal{T}_{u,t} = \{\tau_1, \tau_2, \tau_3 ... \tau_n\}$, where $\tau_1, \tau_2, \tau_3, ..., \tau_n$ refer the occurrence of tag t.

Question 2. How different types of tags are used?

We investigate this problem from many perspectives. In the previous section, we simply assume a tag with high frequency as an indication of a long-term interest of a user. However, we are not sure whether this is correct as there is no temporal information attached to the frequency. Now, we first explore this question by using a simple **temporal activeness** measure of tag t for user u, denoted as $TA_{u,t}$. Let, life scope of tag t, denoted as $LS_{u,t} = \mathcal{T}_{u,t}.\tau_n - \mathcal{T}_{u,t}.\tau_1$, is defined as the time interval between the latest time and the earliest time in $\mathcal{T}_{u,t}$ and, LS_u be the total life scope of the user u in the dataset. Then, we define the temporal activeness $TA_{u,t}$ as follows:

$$TA_{u,t} = \frac{LS_{u,t}}{LS_u}$$

To be meaningful, we rule out tags whose observation period is less than 150 days. According to above equation, we expect to figure out the occurrence pattern of tags. The X axis in Figure 5.4 represents the temporal activeness of all tags for all users. We have divided the ratio equally into 10 groups based on their values from 0.1 to 1.0. Y axis denotes the portion of tags which fall into the corresponding range. Similar as before, we take the frequency into account and compare the results. As shown in the figure, when the tag frequency is low, its temporal activeness is comparatively small. Although it looks very straightforward, *it illustrates that tags of low frequency usually concentrate in a short period. After a certain time, users will hardly use it again which means the short-term interests shift rather fast. Integrating with the high frequency tag's results, we find that long-term interests tend to last for a while instead of concentrating in a short period.*

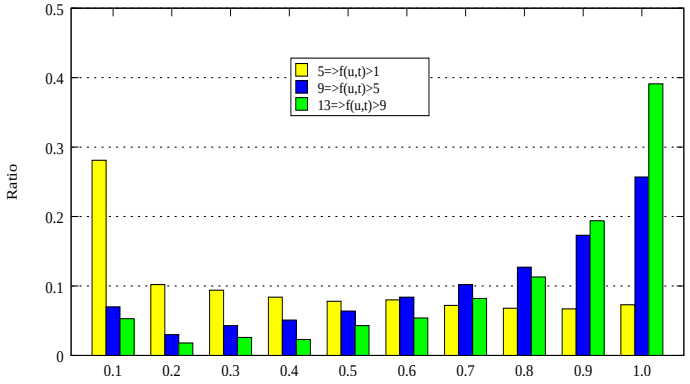


Figure 5.4: Temporal Activeness Distributions of Tags

Although the temporal activeness presents us a brief idea of how the tags distribute in the temporal dimension, we are still not clear of the exact distributions of tags with respect to its temporal usage. More specifically, we are interested to discover the tag's reoccurrence interval and the preference interval. The **reoccurrence interval** represents the average interval between two occurrences of the same tag, whereas **preference interval** captures the time period during when the user has the highest preference for a tag. In another word, it is the most intensive period of the tag usage. Both knowledge is useful for predictions: the reoccurrence interval tells when the tag occurs again and the preference interval tells the most useful temporal duration for predicting the tag.

In order to capture the reoccurrence interval information, we study temporal intervals between the adjacent occurrence of a tag. Given a time series $\mathcal{T}_{u,t}$, the average recurrence interval for a tag t is defined as:

$$RI_{u,t} = \frac{\sum_{i=1}^{n-1} |\tau_{i+1} - \tau_i|}{|\mathcal{T}_{u,t}| - 1}.$$

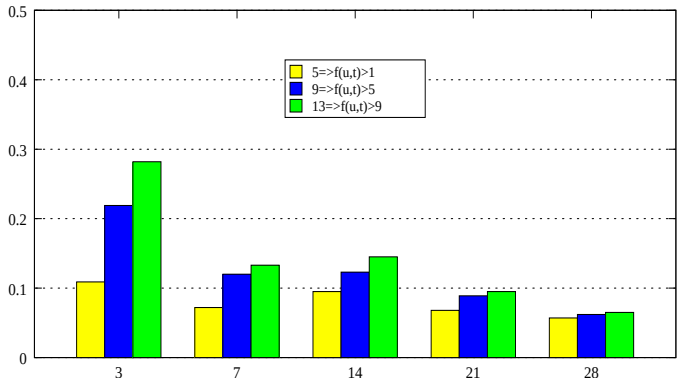


Figure 5.5: Reoccurrence Interval Distributions of Tags in Different Scales

To reveal the underlying patterns of recurrence intervals, we do not simply compute the average intervals between tag occurrence as the standard deviation of the results is quite large, which would make the results less reliable. Instead, we pre-set a number of ranges, such as 3 days, 7 days or etc. We compute the ratio of intervals which fall into the given range compared to all. As shown in Figure 5.5, a comparatively large portion of tags have a small interval. This phenomenon is clearer especially when the tag has a larger frequency. However, these numbers decrease while the range becomes larger. This suggests that when the most recent tag is used a while ago, then it is highly unlikely to be mentioned again by the user. The conclusion we got in this analysis is that *recently used tags have a higher probability to occur again*.

Now we turn to another question, whether there is a peak period during which a tag occurs intensively. In order to capture the preference interval, we do not simply use a ratio measure but adapt a time threshold based clustering algorithm to group individual tag occurrences into temporal clusters. Given $\mathcal{T}_{u,t}$ and T_δ as a given time length, the algorithm puts two points τ_i and τ_j from $\mathcal{T}$ in the same cluster if $|\tau_i - \tau_j \leq T_\delta|$. We initialize a cluster with the first instance in $\mathcal{T}_{u,t}$, and then check whether the followings would be in the same cluster according to the above criteria. If so, we insert it into the existing cluster. Otherwise, we will start a new cluster and repeat above procedures until the last instance.

As an example of the proposed method, let $\mathcal{T}_{u,t}$ = {1,2,3,7,8,10} and T_δ = 1, we have $Cls_{\mathcal{T}_{u,t}}$ = {{1,2,3},{7,8},{10}}. When T_δ = 2, we have $Cls_{\mathcal{T}_{u,t}}$ = {{1,2,3},{7,8,10}}. Each cluster Cls in $Cls_{\mathcal{T}_{u,t}}$ represents a time window during which the tag is valid. We use the first timestamp Cls_{start} of cluster Cls as the start time of the cluster. The clusters in $Cls_{\mathcal{T}_{u,t}}$ are ordered and indexed by their start time. Note that there will be no overlapping between any two clusters. Now, we extract several features from $Cls_{\mathcal{T}_{u,t}}$ as follows:

Table 5.2: Features of Clusters

Freq. NO.	$5 => f > 1$	$9 => f > 5$	$13 => f > 9$
$\|Cls_{\mathcal{T}_{u,t}}\|$	0.82	0.65	0.58
$AvgSize(Cls_{\mathcal{T}_{u,t}})$	1.37	1.96	2.36
$RI(Cls_{\mathcal{T}_{u,t}})$ (days)	109	67	50
$\|Max(Cls_{\mathcal{T}_{u,t}})\|$	1.50	3.04	4.26
$Ind(Max(Cls_{\mathcal{T}_{u,t}}))$	1.02	2.06	2.73
$SC(Cls_{\mathcal{T}_{u,t}})$	0.08	0.55	1.04

- $|Cls_{\mathcal{T}_{u,t}}|$ is the total number of clusters in $Cls_{\mathcal{T}_{u,t}}$.
- AvgSize($Cls_{\mathcal{T}_{u,t}}$) is the average cluster size of $Cls_{\mathcal{T}_{u,t}}$. It can be computed as $\frac{\sum_{Cls \in Cls_{\mathcal{T}_{u,t}}} |Cls|}{|Cls_{\mathcal{T}_{u,t}}|}$,
  where $|Cls|$ denotes the number of points in cluster Cls.
- RI($Cls_{\mathcal{T}_{u,t}}$) is the average reoccurrence interval between adjacent clusters.
- Max($|Cls_{\mathcal{T}_{u,t}}|$ is the point size of the largest cluster in $Cls_{\mathcal{T}_{u,t}}$.
- Ind(Max($Cls_{\mathcal{T}_{u,t}}$)) is the index of the largest cluster in $Cls_{\mathcal{T}_{u,t}}$. In case of a tie, we select the earliest one.
- SC($Cls_{\mathcal{T}_{u,t}}$) is the number of significant clusters in $Cls_{\mathcal{T}_{u,t}}$, where the significant cluster is defined as a cluster with more than two points.

In our analysis, we set T_δ = 7[3] and obtain values of above features for each user tag. In Table 5.2, we have shown results of these features. To compare, we normalize $|Cls_{\mathcal{T}_{u,t}}|$ by its corresponding tag frequency. We observe that $|Cls_{\mathcal{T}_{u,t}}|$ and AvgSize($Cls_{\mathcal{T}_{u,t}}$) increase while the frequency becomes larger. This indicates that many adjacent occurrences of frequent tags are merged in the same cluster. Thus, there will be a more obvious "peak hour" for tags of high frequency. In addition, this number is also in accordance with the results for RI($Cls_{\mathcal{T}_{u,t}}$). Two instances of tags will merge more easily when the interval between them is shorter.

Next, we try to find when the cluster will have the most instances. Such a period indicates the high activeness of a tag. The relatively large number for tags of low frequency once again illustrates those tags are likely to concentrate in a small period. However, for tags of high frequency, they usually have a slot of intensive usage instead of evenly

[3]We have tested several scales T_δ, the trends shown in the results do not change very much. Therefore, we only display results when T_δ equals to 7.

distributed. Next, we expect to find when such a period occurs. This is measured by $Ind(Max(Cls_{\mathcal{T}_{u,t}}))$. We observe that the largest (in term of size) cluster always forms when users begin to use the tag. This is consistent with our common sense. Therefore capturing first several clusters is very meaningful to make good predictions. However, we should note that with the frequency increases, the "peak hour" comes later. This may be explained as for long-term interested tags, users may spend some time to get more familiar with it and then truly accept it as an interest. Except the single largest cluster, we examine the significant clusters [73] to verify the observation from a more general view. The significant clusters are defined as those clusters which contain more than 2 instances. The number of $SC(Cls_{\mathcal{T}_{u,t}})$ reveals that only a few tags have significant clusters. Even if there is, most of them only have one significant cluster on average. It suggests the there are not many "peak hour" for a tag.

In summary, we conclude that, **generally long-term interests get attention all the time instead of from time to time. Second, intervals between active tags are usually small and the most recent tags will probably be used again. Finally, tags tend to have "peak hour" of intensive usage. However, there are not many "peak hours". Therefore, capturing the "peak hour" is very important for tag prediction**.

5.2.3 Correlation Analysis

Following the method from previous sections, these tags are studied separately. However, the same user would probably use similar tags. In this case, each of them could be viewed as a signal for each other. For example, if two tags always occur one after another, then when the first tag appears, it means the second one will appear soon. Except the relatively straightforward co-occurrence information, another factor that can not be ignored is the topic relations between tags. For example, a user is interested at *"Manchester United Club"* and his recent used tag is "Wayne Rooney", then highly probably he will use "Man United" in the future since they are about the similar topic. Therefore, in this section, we expect to study the correlations between tags and check how strong the prediction power.

Question 3. Whether there exist correlations among the tag occurrence? & How to find the correlations?

To achieve the correlation analysis, we propose to construct a context for each tag. The

context is assumed to contain all the tags which have potential positive correlations with the targeting tag. We define the context as: $C_{u,t} = \mathcal{C}(T_u, t)$ & $Context(u, t) \subseteq T_u$

$\mathcal{C}$ is the function which is used to generate the context. In the following section, we first present and compare three algorithmic approaches to automatically discover the context for each tag and then perform correlation analysis among tags and their context.

1. *Naive Context Discovery Method.* The most straightforward method of constructing the context is that we assume all the tags from the same user are the context of each other.

$$\mathcal{C}(T_u, t) = \{T_u \backslash t\}.$$

   However, this assumption is too strong and deviate from the reality. For example, a user may use various tags of different perspectives, such as "football", "basket ball" or "algorithm". Intuitively, the tag "football" would have a much stronger correlation with "basketball" compared to "algorithm". Thus, we present above as the baseline and give another two methods for discovering more accurate context. In those two methods, the assumption is that a tag will be correlated only with part of all the other tags.

2. *Co-occurrence based Context Discovery Method.* In this method, we utilize the co-occurrence information among tags. The idea is that correlated tags are highly probably to be used to annotate the same resource. In particular, in the delicious data, it means they will occur together in the same bookmark. Then, the similarity between two tags is defined as

$$\sigma(t_i, t_j) = \frac{B(t_i) \cap B(t_j)}{B(t_i) \cup B(t_j)} \tag{5.1}$$

   where $B(t_i)$ denotes the set of bookmarks which are annotated by t_i. Similar similarity function can be applied to the bookmark set. We perform an agglomerative clustering to group related tags together. To add more semantics, bookmarks and tags are alternately clustered in the same way until the similarity scores drop below a pre-set threshold. Finally, a set of clusters which include at least two tags are produced. Tags in the same cluster consist the context for each other. More details of this method could be referred in [94]. We utilize the threshold as suggested in that paper.

   We observe that the co-occurrence based method can only find correlated tags that are used together. However, in many cases, there does not have enough co-

occurrence information to cluster them well. To further make an improvement, we propose to use Wikipedia as semantic enhancements to learn the tag's context.

3. *Wikipedia based Context Discovery Method.* Except the little co-occurrence information, another possible correlation that can not be ignored is the topic relations between the tags. The topic information could make up the shortage of occurrence information of many tags.

   We utilize Wikipedia to obtain the topic information of tags. We mainly leverage the techniques proposed in previous chapters to help us to handle the problem. Now, we give a short review of the process. Given a set of tags T_u, first we map each of them to its corresponding Wikipedia concept. This is done with the assist of our pre-built Wikipedia database. More details are in Section 3.2 of Chapter 3. During the mapping process, the ambiguity problem of tags could be eased. For example, given a tag "Apple", by using his historical data, we are able to figure out whether its correct mapping concept should be "Apple Inc." or "Apple" (the fruit). This could help us to capture the user intent more precisely. Consequently, for each user, we collect all the concepts mapped by his tags. Taking the concepts as input, we run the algorithm proposed in Section 4.3 of Chapter 4 and generate a number of clusters (subtrees) according to the Wikipedia knowledge graph.

We apply the last two methods on our dataset and present statistics of the results in Table 5.3. Note that not all the tags could be associated with a cluster in the last two methods. We display the ratio of tags which can be assigned to a cluster in the last column. As shown in Table 5.3, for each user, the Wikipedia based method could find nearly 7 times more clusters compared to the co-occurence method. Meantime, each cluster produced by the Wikipedia method is relative larger.

Table 5.3: Comparison of The Two Methods

	Avg Cluster No.	Avg Cluster Size	Coverage Rate of Tags
Co-occurrence Context	4.65	2.58	42%
Wiki Context	28.69	4.51	71.9%

Given the produced clusters from above methods, we construct a context for each tag by using the other tags which are assigned in the same cluster. Next, we need to figure out whether there are correlations between a tag and its context. We compute the correlation using the statistical measure named Ripley K-function [32]. The K-function is an analysis

method used for event sequences. It can capture the lag relations between events and the results are easy to understand. Assume, the tag t of user u has a context $C_{u,t}$ with k tags $\{t_1, t_2, t_3, ..., t_k\}$ and we want to verify wether tag t is correlated with context $C_{u,t}$ or not. To compute, we first obtain $\mathcal{T}_{u,t}$ and $\mathcal{T}_{u,C_{u,t}}$, where $\mathcal{T}_{u,t}$ is a time series for tag t, and $\mathcal{T}_{u,C_{u,t}}$is an aggregation of a number of time series of tags in $C_{u,t}$. According to the K-function, we compute the dependence between occurrences of $\mathcal{T}_{u,t}$ and $\mathcal{T}_{u,C_{u,t}}$ as follows:

$$\widetilde{K}_{u,t,C_{u,t}} = \frac{|\mathcal{T}_{u,t} \cup \mathcal{T}_{u,C_{u,t}}|}{|\mathcal{T}_{u,t}||\mathcal{T}_{u,C_{u,t}}|} \sum_{r_i \in \mathcal{T}_{u,t}} \sum_{r_j \in \mathcal{T}_{u,C_{u,t}}} I(|r_i.\tau - r_j.\tau| < \beta) \tag{5.2}$$

where $r_i.\tau$ and $r_i.\tau$ are timestamps in $\mathcal{T}_{u,t}$ and $\mathcal{T}_{u,C_{u,t}}$ respectively, and $I()$ is the identity function. β is a pre-set threshold. As implied in the K-function, the values of $\widetilde{K}_{u,t,C_{u,t}}$ greater than $2 * \beta$ suggest there is attraction, or synchrony, between t and $C_{u,t}$, within a window of β. The value of $\widetilde{K}_{u,t,C_{u,t}}$ close to $2*\beta$ suggests no relationship or independence between $\mathcal{T}_{u,t}$ and $\mathcal{T}_{u,C_{u,t}}$, and the value less than $2 * \beta$ suggests there is repulsion, or asynchrony between the tag and its context.

In the remaining section, we analyze results of different context discovery methods. The evaluation goes as follows: we compute $\widetilde{K}$ value for every tag and its context. If the value is larger than $2 * \beta$, we take it as a positive correlation between it and its context. The correlation ratio in Table 5.4 is computed as:

$$\frac{\sum_{t \in T_u} |\widetilde{K}_{u,t,C_{u,t}} > 2 * \beta|}{|T_u|}$$

This is used to examine how many tags are correlated with its context. We also record the $\widetilde{K}$ value in the "correlation significance" column as the value suggesting the significance of correlations. As shown in Table 5.4, although we only select part of the total tags as the context for a tag, our method achieves the best performance in both the ratio and significance evaluation. This indicates that tags of similar topics have larger correlations than the others. The reason why our method outperforms the first one lies in that, if a tag's context is not selected properly, more noise will be introduced in the correlation computation, then the correlation will be negatively effected and the performance will be degraded. The relative lower results for the second method illustrate that simply counting on co-occurrence information is not so reliable as the information is not complete for many tags.

The conclusion in this section is that **there exist correlations between certain tags and**

Table 5.4: Correlation Test

	Correlation Ratio	Correlation Significance
Naive Context	31.67%	33.59
Co-occurrence Context	47.18%	49.07
Wiki Context	77.24%	61.70

we are able to find their correlated tags. But it only works for a limited number of tags due to various constraints.

5.2.4 Social Influence Analysis

Intuitively, the behavior and interests of a user are believed to be influenced by their neighbors in social networks. Many works [79, 5] have proposed methods to qualitatively measure the existence of social influences. In [97], they further attempt to infer users' interests from their social neighbors. In these works, the assumption is that a user will share similar interests with his neighbors. However, we doubt whether this assumption is valid in the tag prediction environment. Moreover, what will happen when we attach more constraints on the influence, such as sources, direction or temporal information.

Question 4. Are there social influences on the tag usage of a user, especially when there exist constraints?

First, we present a brief qualitative study on the data. Out of the 1867 users in the dataset, 1861 have a non-empty neighbor set with an average size as 8. We compute the overlapping of a user's tags with his neighbors by the following equation.

$$TNF(u, F_u) = \frac{|T_u \bigcap T_{F_u}|}{|T_u|}$$

where, T_u is the set of tags for user u and T_{F_u} are the tags from his neighbors F_u. It turns out that on average, 34.3% of user tags could find a corresponding usage in at least one of his neighbors. As a comparison, we compute the above score between a user and a number of randomly chosen users with the same size as F_u. We find that the score drops to 24.9%. This verifies that neighbors indeed share more tags with the targeting user. Furthermore, the shared tags have an average frequency of 2.51 compared to 1.73 of the others.

In above equation, we count the number when there is a matching. However, the matching can not be easily ascribed to the influence. One of the reason is that the influence is directional, which means a user could distribute or receive influences from social neighbors. Therefore, we further attach directional information to the computation. Similar as in [76], we define a tag is *imported* if the user uses it after at least one of his neighbors. Based on the imported tag definition, we compute the network dependence and favorite neighbor dependence of each user using equation 5.3 and equation 5.4 respectively.

$$ND(u) = \frac{|\text{Imported Tags of user u}|}{|T(u)|} \tag{5.3}$$

A user is **network dependent** if most of his tags are imported, i.e, $ND(u)$ is very high. In Figure 5.6(a), we have shown the distribution of users w.r.t. their network dependence. We equally separate all the users' ND scores into 10 histograms from small to large. The Y axis represents the portion of users whose ND scores fall in the corresponding range. We observe that a significant portion of users are network independent (the leftmost histogram) while there also exist many users who are network dependent (rightmost). This suggests that there is no unique conclusion for this problem but have to study case by case.

For network dependence, we further check whether it is because of a single favorite neighbor or the whole neighborhoods. Here we define favorite as how similar of the two users.

$$Sim(u, u') = \frac{T_u \cap T_{u'}}{T_u \cup T'_u}$$

Then a user is **favorite neighbor dependent** if most of his tags are imported from his most favorite neighbor, i.e, $FD(u)$ is very low. As shown in Figure 5.6(b), we can observe that a majority of users adopt tags only from one or two users. This suggests that compared to the whole neighborhood, users tend to be influenced by his favorite neighbors.

$$FD(u) = \frac{|\text{neighbors from whom user u has imported tags}|}{|F_u|} \tag{5.4}$$

Note that, so far the temporal property of social influence is not properly utilized in previous definitions. For example, if a user mentions a tag soon after it occurs in the neighborhood, the influence is comparatively more reliable. Therefore, we further examine the *temporal* property of the tag influence. Within the temporal constraint, users have imported tags from neighbors only if there exists at least one neighbor who has mentioned the same tag earlier than a given time threshold. According to the temporal influence prop-

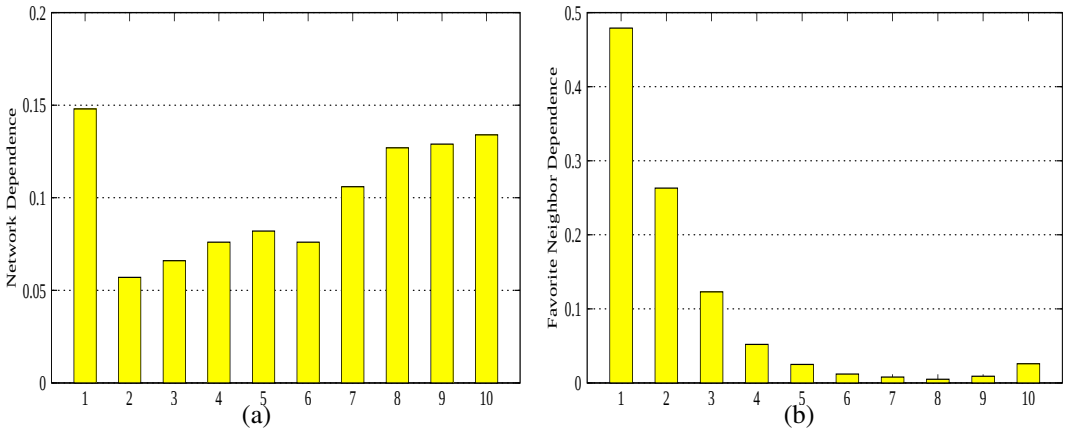


Figure 5.6: User Distribution for Network Dependence and Favorite Neighbor Dependence

erty, we define two measures named utilization rate and popularity rate for each tag. The *utilization rate* of imported tag t for user u is the number of times t has been mentioned by neighbors in a certain time range, whereas, the *popularity* of imported tag t is the number of neighbors who meet the above qualification.

Table 5.5: Temporal Analysis on Social Influence

Time Threshold	Ratio of Imported Tags within Time Threshold	Avg. Utilization Rate	Avg. Popularity Rate	Tag adopted from Favorite Neighbor
3	20%	1.85	1.21	45.1%
7	25%	1.95	1.23	49.3%
14	32%	2.16	1.25	52.4%
30	44%	2.52	1.29	55.7%
60	55%	3.0	1.34	57.5%
Infinity	100%	4.8	1.56	63.5%

In Table 5.5, we display a set of findings given different time interval thresholds. The first column is the time threshold we set in different scales. It expresses the time interval between the first occurrence of a tag for a user and the nearest usage from his neighbors. The second column is the percentage of imported tags within the given time threshold compared to all imported tags. As seen from Table 5.5, if there is no restriction, the average interval equals to 160 days which is a relative long period. In this case, it is hard to determine whether the user accepts the tag due to the social influence or not. When the imported tags are accepted within a shorter interval, the influence makes more sense. We observe that, users adopts 25% of their imported tags within only 7 days. In another word, if a user is network dependent, he will have a higher preference of those tags which are reported recently. The utilization rate and popularity are displayed in the third and forth column. We observe that the utilization rate is consistently larger than that of the

popularity rate. This suggests that, user tends to adopt the tags that are mentioned multiple times, even those references are from the same user. Finally, the last column in Table 5.5 suggests that most of the tags are adopted from his favorite neighbors instead of the whole neighbor. For example, the last number denotes that 63.5% of imported tags are from the same user, his most favorite one. The number shown here is consistent with Figure 5.6(b).

The conclusion in this section is that **when there are more constraints, the social influence becomes more reliable while the effect becomes weak. Generally, the social influence is not so obvious. Compared to the whole neighborhoods, users tend to receive influences only from his favorite neighbor. In addition, the social influences are more significant for certain users than the others. Therefore, it should be considered case by case.**

5.3 Experimental Study

In previous sections, we have analyzed user tags from different perspectives. Now, we experimentally show that these discovers could be directly applicable to user tag prediction problem.

5.3.1 Feature Extraction

To build the personalized prediction model for user u, we construct one feature vector for each tag in T_u using information from his profile. We introduce the extracted features as follows:

1. *Frequency Feature (FF):* As introduced in Section 5.2.1, we use the tag frequency as the first feature. This is also the most widely used feature in many works.

2. *Temporal Feature (TF):* Based on the time series data of tag $\mathcal{T}_{u,t}$, we first extract 3 features named the trend, the occurrence of the tag in the recent time period compared to its whole life and last occurrence interval, the time length from the last occurrence to now. Next, we generate a set of temporal clusters $Cls_{\mathcal{T}_{u,t}}$ by applying the introduced clustering method on $T_{u,t}$. Using $Cls_{\mathcal{T}_{u,t}}$, we extract 7 features which are the cluster number, average cluster size, the size of the last cluster in $Cls_{\mathcal{T}_{u,t}}$, temporal intervals between clusters, the time length of last cluster to now, the index of the largest cluster so far.

3. *Correlation Feature (CF):* First, we need to build the context to find correlations of tags. As shown in Section 5.2.3, the Wikipedia based method achieves the best performance. Therefore, we build the context for each tag according to that method. Since we have assigned the topic information to the tag. The whole context could be viewed as a single topic. Then the topic popularity, which is the total tag frequency of all tags in the context and the size of the context are significant features to express user interests. If a user is interested at a particular topic, he may prefer using similar tags for the topic. Another feature we obtained is the lag value from K-function. Next, we obtain the aggregated time series data of the context, i.e., $\mathcal{T}_{u,C_{u,t}}$, which is the combination of all time series data of each tag in the context. From $\mathcal{T}_{u,C_{u,t}}$, we also extract 9 features discussed in perspective 2.

4. *Social Feature (SF):* Finally, we utilize information from users' neighbors to build the social features. For tag t, we discover the popularity of it in the neighborhood which is computed as: $POP(t) = \sum_{u' \in F_u} T_{u,t} * Sim(u, u')$. Then we calculate the network dependence of the user and whether the tag is shared with his favorite neighbor. Additionally, we obtain an aggregated time series as before. But instead of context tags, we use the tag usage information from his neighbors to build the time series data and extract the features.

We summarize above features in Table 5.6. Our method is a combination of all the features, named "ALLF". Note that not all tags have the whole listed features due to various reasons, including some of them may not have contexts or they are not shared in the neighborhood. We only generate corresponding features to those applicable cases.

Table 5.6: Features from Different Perspectives

Feature Name	Feature Number
Frequency	1
Temporal	9
Correlation	12
Social	12

5.3.2 Experimental Setup

We perform the prediction only on active users, which means he must have used more than 20 tags. Furthermore, We restrict our attention only to tags which have valid data on at least one of features above. Otherwise, it is meaningless.

Next, we need to build the ground-truth dataset to validate our results. The ground-truth data is obtained from the Delicious dataset introduced before. For each user, we do not use the whole records but only the information available up to a given amount time. This is referred as our training data. Then we retrieve a fixed length data from the remaining as the validation data. We build the user profile and extract features from the training data. Given a tag, we verify whether it occurs or not in the validation data. If it occurs, we take it as a positive example. Otherwise, the negative example. We attach the verified results as labels to the feature vectors. Then the prediction problem is transformed as the binary classification problem. We normalize all the features and employ the classifier from Weka[4], a widely used machine learning tool to handle the task. Before the classification, we perform the feature selection by using the method provided by Weka. During the experiment, we have testified and compared several classifiers, including J48, SVM, Logistic and Adaboost. These classifiers share similar performance and reach the same conclusion. Therefore, we only display the results of J48. We perform a 10-fold cross validation on the user tags and report the classification results.

Since we aim on predicting what tags will be used as well as which will not. Therefore, the evaluation should consider both of the two perspectives. For each perspective, we use the widely used evaluation scheme, F-measure to evaluate the prediction results. The final F-measure is a linear combination of both the positive and negative examples.

5.3.3 Comparison Method

As mentioned before, most of current works focus on resource tag prediction. The most similar work is the method proposed in [101], named $MS - IPF$. $MS - IPF$ builds a session-based temporal graph (STG) according to the information existing between user, tags and time. It predicts user tags based on the graph. Although it is designed for the Top-N recommendation task, it can be easily adapted to our problem setting. In the experiment, we build the STG according to the training data and take the top-N results as the positive examples, the bottom-N as the negative ones. Then we check whether the prediction is correct or not according to the validation data. We have tested several possible N and choose the optimal results with N equals to half of the tag size.

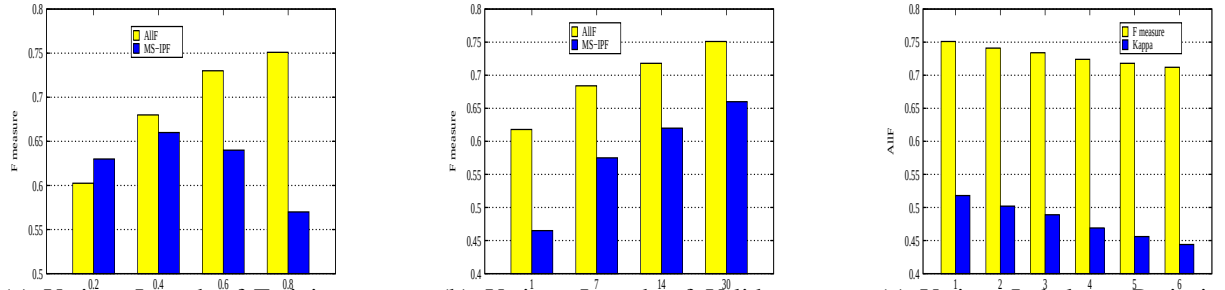
Another set of comparison methods are the variants of our method. In these variants, the difference is in the feature list. Instead of using all the features as $ALLF$, we may only

[4]http://www.cs.waikato.ac.nz/ml/weka/

choose one perspective when building the feature vectors. In this way, we aim to evaluate the effectiveness of different proposed features.

5.3.4 Parameter Analysis

There are many factors which may influence the prediction results. In the following, we investigate the impacts of them and compare the F-measure.



(a) Various Length of Training Data

(b) Various Length of Validation Data

(c) Various Imbalance Ratio in Training Data

Figure 5.7: Different Parameter Setting

Training Data Length TR_l.

One significant issue in recommendation tasks is the "cold start problem". It is caused by the scarcity of available data. Similarly in our case, the length of training data determines how much knowledge we could obtain for building features for tags. In this part, we investigate the influence of training data length on the prediction results. Since the length of user records is different, we are not going to set a absolute length here but using a relative length. We fix the length of validation data as 30 days and compare the performance of various TR_l. In Figure 5.7(a), the X axis denotes the various length of training data compared to the total length of the whole record. As shown in the figure, $MS - IPS$ has a better result for less training data. We explain this as at the beginning, we do not have enough knowledge to build high quality features. Then the performance will be hurt. However, when the length is longer than a certain time period, our method will consistently beat $MS - IPS$. Note that when the ratio is larger than 0.4 (approximately 50 days), our method outperforms $MS - IPS$ even it has more data. Another trend shown in the figure is that $MS - IPS$ has a peak in its performance. After checking the detailed results, we find that while the ratio becomes larger, the F measure for positive predictions

decreases but increases for negative predictions. This is because when the training data becomes larger, the truly positive examples in the validation data will become smaller. However, since we fix the positive example number in the evaluation, the precision for positive cases will decrease. Opposite phenomena happens for negative examples. At last the combination of two F measures will reach its peak when there is a balance.

Validation Data Length VA_l

The second parameter we expect to examine is the length of validation data VA_l, which suggests how long the prediction will keep valid.We display the results in Figure 5.7(b). As suggested in the figure, our method outperforms $MS - IPF$ in all the settings. Besides, both of the two methods perform better when there is more validation data. One reason of this is we will have more test instances when there is more validation data. Another reason is that the user tag prediction is more valid for a longer period. It is hard to require the predictions become true at once.

Class Imbalance Ratio

The user record will evolve while the time goes. Therefore, the number of positive and negative examples will vary when building the prediction model. Generally, when the users' record becomes larger, as most tags only appear quite few times, the negative examples will become many more compared to the positive examples. Thus, it will lead to the class imbalance problem and learning is particularly hard in domains with high class imbalance. We define the imbalance ratio as $IR = |NegativeExamples|/|PositiveExamples|$. To examine the influence of the class imbalance problem, we manually control the imbalance ratio in the experiment and show the results in Figure5.7(c). Besides F measure, we also employ $Kappa$ in the evaluation. $Kappa$ is a chance-corrected measure of agreement between the classifications and the true classes. It is calculated by taking the agreement expected by chance away from the observed agreement and dividing by the maximum possible agreement. A value greater than 0 means that the classifier is doing better than chance. As shown in the experiments, although both of two measures decrease while the imbalance becomes serious, the decrease is not so significant and the scores remain in a relative high positions. Therefore, we conclude that the our method is class imbalance tolerant.

5.3.5 Overall Performance Comparison

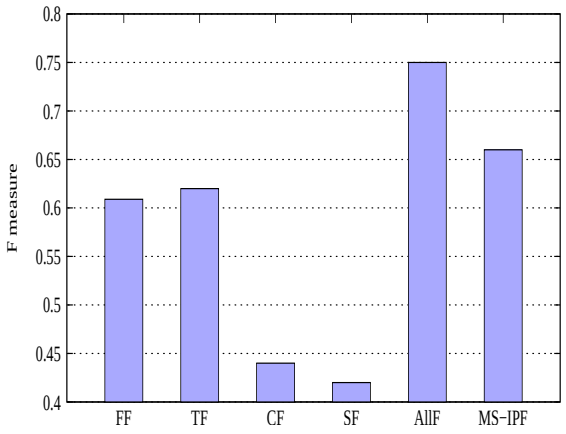


Figure 5.8: F measure for Different Methods

In this section, we compare the overall performance of all algorithms in the prediction task. All of them are under their optimal parameters.

Figure 5.8 shows the F measure comparison for all the methods. First, for the single perspective feature based methods, the first four, TF has the best F measure while SF has the lowest score. This illustrates that incorporating temporal information is effective for predictions. Moreover, the first two have a better score than the last two. This demonstrates that since the applicability of the last two is restricted, using them alone will not get good results. They act more as complementary features to improve the results. Features of different perspectives may be more effective in different situation. Therefore, when combine them together, the results will be improved to a large extent. By using TF as the benchmark, $AllF$ gets a performance gain of 20.9% .

Finally, comparing $AllF$ with $MS - IPF$, $AllF$ achieves a 13.7% increase over $MS - IPF$ on the dataset. This proves that the general feature framework captures more important factors in determining future user tags. In addition, it also suggests that personalized supervised method should be considered for prediction tasks.

5.4 Conclusion

Predicting user tags has many real world applications in social networks, e.g., recommendation, advertisement and etc. In this chapter, we have investigated the usefulness of various sources of information for effective tag prediction. We present a prediction model

that considers frequency, temporal, correlation and social influence. We provide an in-depth and comprehensive analysis on the model and many findings are given during the analysis. Our approach is unique in the sense that, we predict the tags for users instead of resources.

In summary, the contributions of our work are listed as follows:

1. Present an in-depth and comprehensive analysis on different perspectives which may influence the usage of tags in the future, including frequency, temporal, correlation and social influences.
2. Propose a unified prediction framework that provides an integrative view of different perspectives.
3. Extract effective features to predict user tags and achieve promising results.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

User-generated content has received more and more attentions recently, due to its user-friendly nature. More in-depth analysis on UGC is required to prompt its future developments. To cater for the demand, in this thesis, we propose several methodologies to enhance the user experience in current UGC services by exploiting their existing data.

First, we present a brand new cross domain search framework across multiple UGC websites, extending the existing annotation method by linking resources to Wikipedia concepts. We develop a Wikipedia-based clustering algorithm to tackle with the challenge of handling noisy tags associated with resources. In this way, resources in different domains, such as documents or images, are represented in the same Wikipedia concept space. Our framework exhibits high extensibility and flexibility on the processing of cross domain search. It only depends on the correlation between resources and Wikipedia concepts. Based on the framework, different types of resources are able to be utilized to describe each other. Therefore, users can get a better idea of the context of the resource. Such a framework will be especially useful considering the heterogeneous attribute of UGC.

Besides integrating resources from different websites, we also propose an alternative interface for users to browse their information flow. In Chapter 4, taking Twitter as an example, we offer a knowledge view for the Twitter stream. The establishment of the interface is inspired from the traditional newspaper where news of similar topics is grouped together. Similarly, we propose to categorize the information flow as well. We achieve the purpose by fully utilizing the Wikipedia contents along with their well-organized hierarchy and high-quality links. In particular, we first map tweets to Wikipedia, which means that each tweet is represented by several Wikipedia concepts. Then we group the transformed tweets by the Wikipedia hierarchy. Finally, we evaluate each constructed group and return them to users in the order of a designed ranking function. With the knowledge view, it is quite easy for users to catch the overview of the stream. They can be guided to the information that they are interested at quickly by following the group labels. According to our user study, users give a positive feedback on the proposed interface.

In the above two works, we mainly focus on explicitly manage and organize resources in current UGC websites. In the last part of the thesis, we intend to explore the usefulness of UGC in user interest modeling and prediction. Taking user tag prediction as an example, we present a unified framework which integrate the frequency, temporal, correlation and social influence information We present an in-depth and comprehensive analysis on the framework. Our approach is unique in the sense that, we model each tag for each user from a series of perspectives. Experimental results on real world dataset suggest that the discovered features are useful to reach a promising performance for future user tag discovery.

6.2 Future Work

How to organize and manage the resources has become one of the most concerned problems in current UGC websites. Besides the works in previous chapters, we plan to follow the next several directions in the future.

6.2.1 Resource Ranking in UGC Websites

Considering the vast amount of information flow in UGC websites, one of the major tasks of these websites is to rank or recommend relevant resources in a meaningful manner so

that users are able to reach the information that they are interested conveniently. However, according to the inherent features of UGC, the ranking becomes significantly different from traditional methods. We list several reasons in the following to explain why traditional methods will not work well in the UGC scenario.

1. *Short Text.* A significant amount of UGC is remarkably short, such as the status or comments in Facebook. In particular, Twitter has a strict rule that requires each tweet must be within 140 words. All these features indicate that compared to expert-edited documents, ordinary users on the Web are inclined to write a relatively short piece of text to express their ideas or feelings. Therefore, the traditional TF-IDF function or the related methods will lose their magic here. When considering the noisy property of UGC, the situation becomes worse.

2. *No clear connections.* One of the success elements of previous methods, such as PageRank, is that there are hyperlinks between web documents. According to these links, the relative importance and connections of documents could be determined. Comparatively, each resource in UGC websites is created independently. There are no explicit links between them. Therefore, it is difficult to assess the importance of each resource.

3. *Query Format.* In conventional resource retrieval, keyword queries are widely used. However, UGC is much more flexible and is expressed in various formats. Correspondingly, to better satisfy users, the format of input queries should be no longer restricted in keywords. For example, user may prefer searching documents by photos taken through their mobile phones. In addition, there also exists an issue that the future system should be able to present users with useful resources even without a query. In this case, it will require the system to understand users in advance.

As illustrated above, these new features bring new challenges on the ranking of UGC. Although we have proposed a cross domain search framework, it is still a long way to go. We believe one possible direction should be data-driven. With the large scale data, we expect data can explain for itself.

6.2.2 A Variety of Visualization Methodologies

Compared to the rapid growth of UGC websites, their interfaces remain the same for a long while. As the website entry, the interface should be able to help users to access the

information effectively. Although existing websites keep modifying their interfaces, such as the "timeline" pushed by Facebook, the style is still in line with its ancestors. However, we argue that more options should be provided to users. In particular, there are two works that can be done:

1. We can develop more visualization techniques to show the information, such as drawing figures according to the data. The questions we need to figure out are what figures to draw and how to draw. We consider that more dimensions could be depicted for the data, such as the spatial, temporal and topic dimensions. From these perspectives, underlying connections between data could be presented to users.
2. We can offer more customized templates for users to manage their resources. Since the data in UGC websites is contributed by users, they should have the right to view the data in their own ways.

6.2.3 User Behavior and Interest Analysis

When users publish a tweet, bookmark a url, upload an image or interact with friends, these activities create profiles for users in UGC websites. On one hand, these profiles provide invaluable information to understand the behavior and interest of a user. On the other hand, these studies bring benefits to online advertising, e-commerce companies and etc. We can provide services on a user by user basis. However, there are several challenges in effectively performing the analysis:

1. *Subtle signals of user interests.* Although it is assumed that user interests are reflected from their personal activities, the traces of interests are usually implicit and hard to track. For example, a user may publish a tweet as "which brand of TV to buy?". Obviously, at that time, the user would be interested at relevant TV advertisements. However, current technologies still can not handle this kind of situations properly.
2. *User behavior and interest prediction.* Existing works focus on identifying user interests from their generated contents. Then the next question will be when will a user become interested at a particular piece of information. Compared to the simple identification, the prediction task is more challenging. Although it looks intractable, we believe that at least there are certain patterns we can discover. Our work in Chapter 5 is a first trial. Next, more data sources and models will be applied to investigate the problem.

BIBLIOGRAPHY

[1] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Semantic enrichment of twitter posts for user profile construction on the social web. In *ESWC*, pages 375–389, Berlin, Heidelberg, 2011. Springer-Verlag. 12, 15

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the State-of-the-Art and possible extensions. *TKDE*, 17(6):734–749, 2005. 7

[3] Amr Ahmed, Yucheng Low, Mohamed Aly, Vanja Josifovski, and Alexander J. Smola. Scalable distributed inference of dynamic user interests for behavioral targeting. In *KDD*, pages 114–122. ACM, 2011. 15, 16, 69

[4] Noor Ali-Hasan and Lada A. Adamic. Expressing social relationships on the blog through links and comments. In *ICWSM*, 2007. 15

[5] Aris Anagnostopoulos, Ravi Kumar, and Mohammad Mahdian. Influence and correlation in social networks. In *KDD*, pages 7–15. ACM, 2008. 82

[6] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: a nucleus for a web of open data. In *ISWC*, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag. 12, 17

[7] John Battelle. Packaged goods media vs. conversational media. 5 December 2006. viii

[8] Michael S. Bernstein, Bongwon Suh, Lichan Hong, Jilin Chen, Sanjay Kairam, and Ed H. Chi. Eddi: interactive topic-based browsing of social status streams. In *UIST*, pages 303–312. ACM, 2010. 6, 14

[9] P. Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337:217–239, 2005. 55

[10] Ulrik Brandes, Patrick Kenis, Jürgen Lerner, and Denise van Raaij. Network analysis of collaboration structure in wikipedia. In *WWW*, pages 731–740. ACM, 2009. 17

[11] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998. 10

[12] Michael M. Bronstein, Alexander M. Bronstein, Fabrice Michel, and Nikos Paragios. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *CVPR*, pages 3594–3601. IEEE, 2010. 11

[13] David Carmel, Haggai Roitman, and Naama Zwerdling. Enhancing cluster labeling using wikipedia. In *SIGIR*, pages 139–146. ACM, 2009. 18

[14] Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Comput. Surv.*, 41(3):17:1–17:38, July 2009. 63

[15] Jilin Chen, Werner Geyer, Casey Dugan, Michael J. Muller, and Ido Guy. Make new friends, but keep the old: recommending people on social networking sites. In *CHI*, pages 201–210, 2009. 43

[16] Jilin Chen, Rowan Nairn, and Ed Huai hsin Chi. Speak little and well: recommending conversations in online social streams. In *CHI*, pages 217–226. ACM, 2011. 6, 13, 14

[17] Jilin Chen, Rowan Nairn, Les Nelson, Michael S. Bernstein, and Ed H. Chi. Short and tweet: experiments on recommending content from information streams. In *CHI*, pages 1185–1194. ACM, 2010. 13

[18] Chirita, Paul Alexandru, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschutter. Using ODP metadata to personalize search. In *Web search*, pages 178–185, 2005. 15

[19] Chris Bizer Christian Becker. Flickrwrappr. In *http://www4.wiwiss.fu-berlin.de/flickrwrappr/*. 17

[20] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979. 54

[21] Steve Cronen-Townsend and W. Bruce Croft. Quantifying query ambiguity. In *HLT*, pages 104–109, 2002. 25

[22] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, pages 708–716. ACL, 2007. 17

[23] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, April 2008. 5, 11, 19

[24] T. Deselaers, D. Keysers, and H. Ney. Features for image retrieval: A quantitative comparison. In *DAGM*, pages 228–236, 2004. 30

[25] Nicholas Diakopoulos, Mor Naaman, and Funda Kivran-Swaine. Diamonds in the rough: Social media visual analytics for journalistic inquiry. In *VAST*, pages 115–122. IEEE, 2010. 14

[26] Marian Dörk, Daniel M. Gruen, Carey Williamson, and M. Sheelagh T. Carpendale. A visual backchannel for large-scale events. *IEEE Trans. Vis. Comput. Graph*, 16(6):1129–1138, 2010. 14

[27] Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and Heung-Yeung Shum. An empirical study on learning to rank of tweets. In *COLING*, pages 295–303, Stroudsburg, PA, USA, 2010. ACL. 13

[28] Xin Fan, Xing Xie, Zhiwei Li, Mingjing Li, and Wei-Ying Ma. Photo-to-search: using multimodal queries to search the web from mobile devices. In *MIR*, pages 143–150. ACM, 2005. 12

[29] Wei Feng and Jianyong Wang. Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In *KDD*, pages 1276–1284. ACM, 2012. 16, 68

[30] Evgeniy Gabrilovich and Shaul Markovitch. Feature generation for text categorization using world knowledge. In *IJCAI*, pages 1048–1053, Edinburgh, Scotand, August 2005. 18

[31] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI*, pages 1606–1611, 2007. 18, 25, 36, 48

[32] Daniel G. Gavin. K1d: Multivariate ripley's k-function for one-dimensional data. University of Oregon, 2010. 80

[33] Ziyu Guan, Jiajun Bu, Qiaozhu Mei, Chun Chen, and Can Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *SIGIR*, pages 540–547. ACM, 2009. 68

[34] Ido Guy, Inbal Ronen, and Ariel Raviv. Personalized activity streams: sifting through the "river of news". In *RecSys*, pages 181–188. ACM, 2011. 13

[35] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *RecSys*, pages 199–206. ACM, 2010. 43

[36] Ben He and Iadh Ounis. Query performance prediction. *Inf. Syst*, 31(7):585–594, 2006. 26

[37] Yasuhide Mori Hironobu, Hironobu Takahashi, and Ryuichi Oka. Image-to-word transformation based on dividing and vector quantizing images with words. In *in Boltzmann machines, Neural Networks*, page 405409, 1999. 12

[38] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *WWW*, pages 229–232. ACM, 2011. 12

[39] Liangjie Hong, Ron Bekkerman, Joseph Adler, and Brian D. Davison. Learning to rank social update streams. In *SIGIR*, pages 651–660. ACM, 2012. 13

[40] Liangjie Hong and Brian D. Davison. Empirical study of topic modeling in twitter. In *SOMA*, pages 80–88. ACM, 2010. 13

[41] Jian Hu, Lujun Fang, Yang Cao, Hua-Jun Zeng, Hua Li, Qiang Yang, and Zheng Chen. Enhancing text clustering by leveraging wikipedia semantics. In *SIGIR*, pages 179–186. ACM, 2008. 17, 18

[42] Jian Hu, Gang Wang, Fred Lochovsky, Jian tao Sun, and Zheng Chen. Understanding user's query intent with wikipedia. In *WWW*, pages 471–480. ACM, 2009. 25

[43] Xiaohua Hu, Xiaodan Zhang, Caimei Lu, E. K. Park, and Xiaohua Zhou. Exploiting wikipedia as external knowledge for document clustering. In *KDD*, pages 389–396. ACM, 2009. 18, 50

[44] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems*, 20(4):422–446, October 2002. 36, 62

[45] J. Jeon, V. Lavrenko, and R. Manmatha. Automatic image annotation and retrieval using cross-media relevance models. In *SIGIR*, pages 119–126. ACM, 2003. 12, 37

[46] Yangqing Jia, Mathieu Salzmann, and Trevor Darrell. Learning cross-modality similarity for multinomial data. In *ICCV*, pages 2407–2414. IEEE, 2011. 11

[47] Khuller, Vishkin, and Young. A primal-dual parallel approximation technique applied to weighted set and vertex covers. *ALGORITHMS: Journal of Algorithms*, 17, 1994. 54, 55

[48] J. Kincaid. Edgerank: The secret sauce that makes facebooks news feed tick. TechCrunch. 6, 13

[49] Shaishav Kumar and Raghavendra Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, pages 1360–1365. IJCAI/AAAI, 2011. 11

[50] Jia Li and James Ze Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Trans. Pattern Anal. Mach. Intell*, 25(9):1075–1088, 2003. 12

[51] Xin Li, Lei Guo, and Yihong Eric Zhao. Tag-based social interest discovery. In *WWW*, pages 675–684. ACM, 2008. 15

[52] Zhenhui Li, Ding Zhou, Yun-Fang Juan, and Jiawei Han. Keyword extraction for social snippets. In *WWW*, pages 1143–1144. ACM, 2010. 14

[53] Huizhi Liang, Yue Xu, Yuefeng Li, Richi Nayak, and Xiaohui Tao. Connecting users and items with weighted tags for personalized item recommendations. In *HT*, pages 51–60. ACM, 2010. 16

[54] Marek Lipczak, Yeming Hu, Yael Kollet, and Evangelos Milios. Tag sources for recommendation in collaborative tagging systems. In *ECML PKDD Discovery Challenge 2009 (DC09)*, volume 497, pages 157–172, September 2009. 16, 68

[55] Nedim Lipka and Benno Stein. Identifying featured articles in wikipedia: writing style matters. In *WWW*, pages 1147–1148. ACM, 2010. 17

[56] Chen Liu, Bing Cui, and Anthony K.H. Tung. Integrating web 2.0 resources by wikipedia. In *ACM Multimedia*, pages 707–710. ACM, 2010. 12, 49

[57] Chen Liu, Beng Chin Ooi, Anthony K.H. Tung, and Dongxiang Zhang. Crew: cross-modal resource searching by exploiting wikipedia. In *ACM Multimedia*, pages 1669–1672. ACM, 2010. 33

[58] Zhiyuan Liu, Xinxiong Chen, and Maosong Sun. Mining the interests of chinese microbloggers via keyword extraction. *Front. Comput. Sci China*, 6(1):76–87, February 2012. 14

[59] Zhongming Ma, Gautam Pant, and Olivia R. Liu Sheng. Interest-based personalized search. *ACM Trans. Inf. Syst*, 25(1), 2007. 15

[60] João Magalhães, Fabio Ciravegna, and Stefan M. Rüger. Exploring multimedia in a keyword space. In *ACM Multimedia*, pages 101–110. ACM, 2008. 12, 19

[61] Adam Marcus, Michael S. Bernstein, Osama Badar, David R. Karger, Samuel Madden, and Robert C. Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *CHI*, pages 227–236. ACM, 2011. 14

[62] Edgar Meij, Wouter Weerkamp, and Maarten de Rijke. Adding semantics to microblog posts. In *WSDM*, pages 563–572. ACM, 2012. 14, 44, 48, 49, 65

[63] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*, pages 233–242. ACM, 2007. 17, 49

[64] David N. Milne and Ian H. Witten. Learning to link with wikipedia. In *CIKM*, pages 509–518. ACM, 2008. 17, 18, 52

[65] Mor Naaman, Jeffrey Boase, and Chih-Hui Lai. Is it really about me?: message content in social awareness streams. In *CSCW*, pages 189–192. ACM, 2010. 13

[66] Monica Lestari Paramita, Mark Sanderson, and Paul Clough. Diversity in photo retrieval: Overview of the imageCLEFPhoto task 2009. In *CLEF*, volume 6242, pages 45–59. Springer, 2009. 5

[67] Jing Peng, Daniel Dajun Zeng, Huimin Zhao, and Fei-yue Wang. Collaborative filtering in social tagging systems based on joint item-tag recommendations. In *CIKM*, pages 809–818. ACM, 2010. 7, 16

[68] Owen Phelan, Kevin McCarthy, and Barry Smyth. Using twitter to recommend real-time topical news. In *RecSys*, pages 385–388. ACM, 2009. 15

[69] Benjamin Piwowarski and Hugo Zaragoza. Predictive user click models based on click-through history. In *CIKM*, pages 175–182, November 2007. 15

[70] Guojun Qi, Charu C. Aggarwal, and Thomas Huang. Towards semantic knowledge propagation from text corpus to web images. In *WWW*, pages 297–306. ACM, 2011. 19

[71] Daniel Ramage, Susan T. Dumais, and Daniel J. Liebling. Characterizing microblogs with topic models. In *ICWSM*. The AAAI Press, 2010. 14, 62

[72] Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert R. G. Lanckriet, Roger Levy, and Nuno Vasconcelos. A new approach to cross-modal multimedia retrieval. In *ACM Multimedia*, pages 251–260. ACM, 2010. 11, 19

[73] Tye Rattenbury, Nathaniel Good, and Mor Naaman. Towards automatic extraction of event and place semantics from flickr tags. In *SIGIR*, pages 103–110. ACM, 2007. 78

[74] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD*, pages 727–736. ACM, 2009. 15, 16

[75] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90. ACM, 2010. 16, 68

[76] Daniel M. Romero, Brendan Meeder, and Jon M. Kleinberg. Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter. In *WWW*, pages 695–704. ACM, 2011. 73, 83

[77] Xiaoguang Rui, Mingjing Li, Zhiwei Li, Wei-Ying Ma, and Nenghai Yu. Bipartite graph reinforcement model for web image annotation. In *ACM Multimedia*, pages 585–594. ACM, 2007. 12

[78] Mehran Sahami and Timothy D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW*, pages 377–386. ACM, 2006. 12, 25

[79] Parag Singla and Matthew Richardson. Yes, there is a correlation: - from social networks to personal behavior on the web. In *WWW*, pages 655–664. ACM, 2008. 15, 82

[80] Frank Smadja. Mixing financial, social and fun incentives for social voting. In *Workshop on WEBCENTIVES*. ACM, 2007. 3

[81] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *workshop on Multimedia information retrieval*, pages 321–330. ACM, 2006. 5

[82] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, December 2000. 5

[83] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell*, 22(12):1349–1380, 2000. 11

[84] Micro Speretta and Susan Gauch. Personalized search based on user search histories. In *WI*, pages 622–628, Washington, DC, USA, 2005. IEEE Computer Society. 15

[85] Julia Stoyanovich, Sihem Amer-Yahia, Cameron Marlow, and Cong Yu. Leveraging tagging to model user interests in del.icio.us. In *AAAI*, 2008. 69

[86] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *RecSys*, pages 43–50. ACM, 2008. 16

[87] Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, pages 449–456. ACM Press, 2005. 67

[88] Diego Torres, Pascal Molli, Hala Skaf-Molli, and Alicia Diaz. Improving wikipedia with dbpedia. In *WWW*, pages 1107–1112. ACM, 2012. 17

[89] Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, September 1995. 58

[90] G. Vickery and S. Wunsch-Vincent. *Participative Web and User-Created Content: Web 2.0, Wikis and Social Networking*. October 2007. 1

[91] S. V. N. Vishwanathan and Alexander J. Smola. Fast kernels for string and tree matching. In *NIPS*, pages 569–576, 2002. 56

[92] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk — A link discovery framework for the web of data. In *2nd Workshop on Linked Data on the Web*, Madrid, Spain, April 2009. 17

[93] Chi Wang, Rajat Raina, David Fong, Ding Zhou, Jiawei Han, and Greg Badros. Learning relevance from heterogeneous social network and its application in online targeting. In *SIGIR*, pages 655–664. ACM, 2011. 12, 15

[94] Haofen Wang, Yan Liang, Linyun Fu, Gui-Rong Xue, and Yong Yu. Efficient query expansion for advertisement search. In *SIGIR*, pages 51–58. ACM, 2009. 79

[95] James Ze Wang, Jia Li, and Gio Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture LIbraries. *IEEE Trans. Pattern Anal. Mach. Intell*, 23(9):947–963, 2001. 11

[96] Pu Wang and Carlotta Domeniconi. Building semantic kernels for text classification using wikipedia. In *KDD*, pages 713–721. ACM, 2008. 18

[97] Zhen Wen and Ching-Yung Lin. Improving user interest inference from social neighbors. In *CIKM*, pages 1001–1006. ACM, 2011. 15, 82

[98] Ryen W. White, Peter Bailey, and Liwei Chen. Predicting user interests from contextual information. In *SIGIR*, pages 363–370. ACM, 2009. 15

[99] Ryen W. White, Paul N. Bennett, and Susan T. Dumais. Predicting short-term interests using activity-based search context. In *CIKM*, pages 1009–1018. ACM, 2010. 15, 67

[100] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *WWW*, pages 635–644. ACM, 2008. 17

[101] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *KDD*, pages 723–732. ACM, 2010. 87

[102] Tom Yeh, Konrad Tollmar, and Trevor Darrell. Searching the web with mobile images for location recognition. In *CVPR*, pages 76–81, 2004. 12

[103] Hilmi Yildirim and Mukkai S. Krishnamoorthy. A random walk method for alleviating the sparsity problem in collaborative filtering. In *RecSys*, pages 131–138. ACM, 2008. 16, 68

[104] Dawei Yin, Liangjie Hong, Zhenzhen Xue, and Brian D. Davison. Temporal dynamics of user interests in tagging systems. In *AAAI*, 2011. 16

[105] Dawei Yin, Zhenzhen Xue, Liangjie Hong, and Brian D. Davison. A probabilistic model for personalized tag prediction. In *KDD*, pages 959–968. ACM, 2010. 15, 16

[106] Oren Zamir and Oren Etzioni. Web document clustering: a feasibility demonstration. In *SIGIR*. ACM, 1998. 57

[107] Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. Tagicofi: tag informed collaborative filtering. In *RecSys*, pages 69–76. ACM, 2009. 16

[108] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *KDD*, pages 940–948. ACM, 2012. 11