University of California

Los Angeles

Challenges and Opportunities in Building Personalized Online Content Aggregators

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Ka Cheung Sia

2009

The dissertation of Ka Cheung Sia is approved.

Carlo Zaniolo

Richard R. Muntz

Christine L. Borgman

Junghoo Cho, Committee Chair

University of California, Los Angeles

2009

To my parents

Table of Contents

List of Figures

List of Tables

Acknowledgments

First of all, I would like to thank my adviser Junghoo Cho for his guidance and support during my days in UCLA. He is always available and eager to help me whenever I need. I have benefited tremendously from his feedback, ideas, and criticism on my research, writing, and presentation skills. I am indebted to him for what I have become as a researcher.

I would also like to thank the members of my committee, Carlo Zaniolo, Christine Borgman, and Richard Muntz for providing valuable comments on my dissertation. I have benefited a lot from them to improve the quality of this dissertation.

Part of this dissertation is the result of fruitful collaborations when I worked in NEC Labs America as an intern. I would like to thank Belle L. Tseng, Yun Chi, Koji Hino, and Shenghuo Zhu for the valuable ideas, discussions, and research techniques I have learned from them.

Many thanks go to Alexandros Ntoulas who helped me in all aspects to start my research work in UCLA and onwards. I would also like to thank my friends who had helped me and I learned from: Jianming He, Hetal Thakkar, Sourashis Roy, Victor Liu, William So, Guoling Han, Yijian Bai, Sungjin Kim, Uri Schonfeld, Susan Chebotariov, Michael Welch, Barzan Mozafari, Albert Lee, Amruta Joshi, Felix Gao, Ray Cheung, and Tierui Chen.

I would like to extend my thanks to Irwin King who had encouraged and helped me to study abroad that started my journey to explore the world of research. Last but not least, I would like to thank all the scientists and engineers for their contributions for us to stand on the shoulders of giants to see and shape our future.

Vita

1979	Born, Hong Kong
2001	B.Eng. in Information Engineering, The Chinese University of Hong Kong, Hong Kong.
2003	M.Phil. in Computer Science, The Chinese University of Hong Kong, Hong Kong.
2004 – 2008	Graduate Student Researcher, Department of Computer Science, UCLA
2006	M.S. in Computer Science, University of California, Los Angeles, CA, USA.
2006,2007	Research Intern, NEC Labs America, Cupertino, CA, USA
2008	Software Engineering Intern, Google Inc., Mountain View, CA, USA
2009	Ph.D. in Computer Science, University of California, Los Angeles, CA, USA.
2009 –	Software Design Engineer, Live Search, Microsoft Corporation, Redmond, WA, USA

Publications

Irwin King, Cheuk Hang Ng, and Ka Cheung Sia. Distributed Content-Based

Visual Information Retrieval System on Peer-to-Peer Networks. *In ACM Transaction of Information System 22(3):477–501, 2004.*

Yu-Ru Lin, Wen-Yen Chen, Xiaolin Shi, Richard Sia, Xiaodan Song, Yun Chi, Koji Hino, Hari Sundaram, Jun Tatemuran, and Belle L. Tseng. The Splog Detection Task and A Solution Based on Temporal and Link Properties. *In Proceedings of the* 15th *Text Retrieval Conference (TREC), 2006.*

Zhengyu Liu, Ka Cheung Sia, and Junghoo Cho. Cost-Efficient Processing of MIN/MAX Queries over Distributed Sensors with Uncertainty. *In Proceedings of the* 20th *ACM Symposium on Applied Computing (SAC), 2005.*

Cheuk-Hang Ng and Ka Cheung Sia. Peer Clustering and Firework Query Model. *In Poster Proceedings of the* 11th *World Wide Web Conference (WWW), 2002.*

Cheuk-Hang Ng and Ka Cheung Sia. Advanced Peer Clustering and Firework Query Model in the Peer-to-Peer Network. *In Poster Proceedings of the* 12th *World Wide Web Conference (WWW), 2003.*

K. C. Sia and Irwin King. Relevance Feedback Based on Parameter Estimation of Target Distribution. *In Proceedings of World Congress on Computational Intelligence, 2002.*

Ka Cheung Sia, Cheuk Hang Ng, Chi Hang Chan, Siu Kong Chan, and Lai Yin Ho. Bridging the P2P and WWW Divide with DISCOVIR - DIStributed COntent-based Visual Information Retrieval. *In Poster Proceedings of the* 12th

World Wide Web Conference (WWW), 2003.

Ka Cheung Sia. Content Based Image Retrieval: Reading One's Mind and Helping People Share. *Master Thesis, The Chinese University of Hong Kong, 2003.*

Ka Cheung Sia. DDos Volunerability Analysis of the Bittorrent Protocol. *Technical Report, University of California, Los Angeles, 2006.*

Ka Cheung Sia, Junghoo Cho, and Hyun-Kyu Cho. Efficient Monitoring Algorithm for Fast News Alert. *In IEEE Transaction of Knowledge and Data Engineering 19(7):950–961, 2007.*

Ka Cheung Sia, Junghoo Cho, Koji Hino, Yun Chi, Shenghuo Zhu, and Belle L. Tseng. Monitoring RSS Feeds based on User Browsing Pattern. *In Proceedings of the* 1^{st} *International Conference on Weblog and Social Media, 2007.*

Ka Cheung Sia, Shenghuo Zhu, Yun Chi, Koji Hino, and Belle L. Tseng. Capturing User Interests by Both Exploitation and Exploration. *In Proceedings of the International Conference on User Modeling, 2007.*

Ka Cheung Sia, Junghoo Cho, Yun Chi, and Belle L. Tseng. Efficient Computation of Personal Aggregate Queries on Blogs. *In Proceedings of the ACM Knowledge Discovery and Data Mining Conference, 2008.*

Abstract of the Dissertation

Challenges and Opportunities in Building Personalized Online Content Aggregators

by

Ka Cheung Sia

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2009

Professor Junghoo Cho, Chair

The emergence of "Web 2.0" services has attracted a large number of users to publish content on the Web as blogs, social bookmarks, customer reviews, and wiki articles. Due to the explosion of this "user-generated" content, the amount of new data on the Web is growing rapidly, at a rate that is several times higher than what was believed before. To help users keep up with the continuous stream of new content, many Web 2.0 sites provide RSS feeds that return recently updated materials based on a user's request. Since many users often "subscribe" to a large number of RSS, which may be on the order of hundreds, there is a need for a new online service that helps users manage their growing subscription lists and the constant stream of new content.

In this dissertation, I study the challenges and opportunities in building a large-scale personalized online content aggregator. In particular, I address the following three challenges in building such a system:

(1) A significant portion of user-generated content is updated frequently, often several times a day, and is related to current world events whose significance deteriorates rapidly over time. I propose an effective RSS-feed retrieval algorithm

based on the updating pattern of the feeds and the access pattern of the users. The algorithm helps the system deliver fresh content to the users in a timely manner even in a resource-constrained setting.

(2) In order to help users navigate the continuously updated new content, it is important to provide a service that can prioritize and recommend what the user is most likely to be interested in based on the user's personal interest. I propose a learning framework that efficiently captures a user's interest through an interactive process. I also develop an efficient approach to computing such "personalized recommendations" that can scale well to a large number of users without putting an unreasonable strain on computing resources.

(3) The data collected from such a system over time, often in the form of social annotations, involves lots of human effort in matching the best descriptive keywords with the corresponding Web resources. As an example of how this rich body of data can be mined to help users, I analyze its evolution over time and propose methods that discover the behavioral properties of evolving vocabulary usage. I illustrate how these properties can be used to help users select the best keywords in the online advertising context.

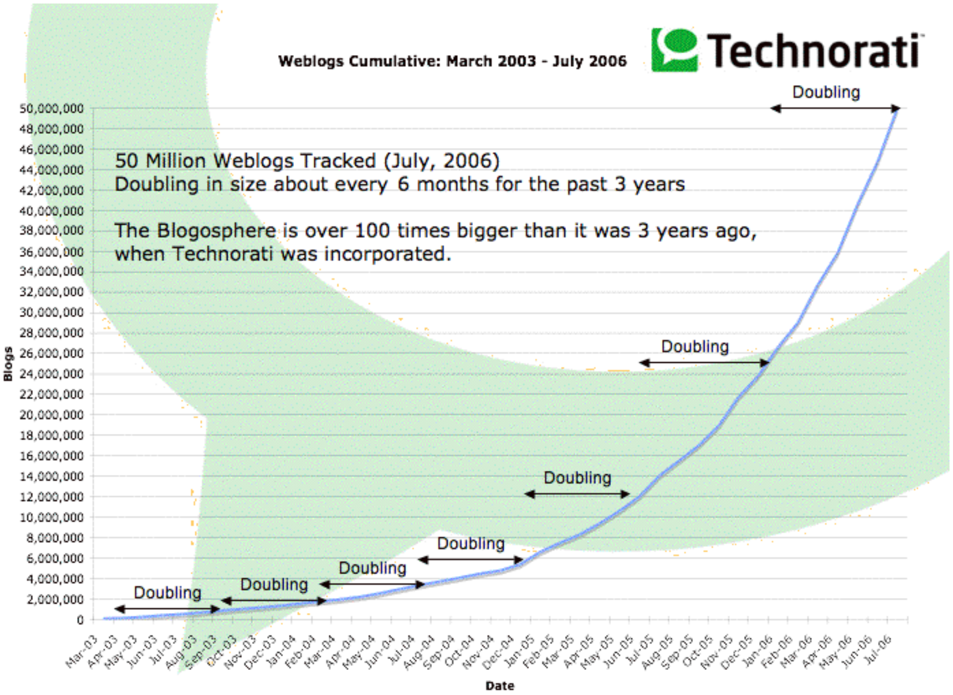
CHAPTER 1

Introduction

The “Web 2.0” services, such as Blogger [Bloa], MySpace, Flickr, and delicious [Del], have made it extremely easy even for non-technical amateurs and individuals to share their content online, resulting in the explosion of new online materials, “user-generated” content (UGC). For example, according to Ramakrishnan and Tomkins [RT07], the rate that UGC is produced is 4 times larger than that of professional Web content (written by somebody paid for this work, such as a corporate site’s Web master). Figure 1.1 shows a study by Technorati, a search engine targeting on blog content, that the number of blogs[1] had been doubling every 6 months for 3 years form 2003 to 2006.

The explosion of online sources of information makes it very difficult for a user to keep up with what is new on the Web. That is, when a user has only a few sources of interest, say, the *New York Times* and CNN, it is reasonable to expect that she will visit these sites regularly to browse through new materials. When she has hundreds of such sites, however, it is practically impossible for her to visit each site regularly to see what is new. To help users in this context, many Web 2.0 sites provide *RSS feeds* [RSSa] for new content. An RSS feed returns recently updated materials at a site according to a user’s request, and because it is based on a standard protocol, an automatic software program, called an *RSS reader*, can perform periodic retrievals of new materials from RSS feeds. An RSS reader,

[1] Only blog creations are counted, blogs that were abandoned are still counted.



50 Million Weblogs Tracked (July, 2006)
Doubling in size about every 6 months for the past 3 years

The Blogosphere is over 100 times bigger than it was 3 years ago,
when Technorati was incorporated.

Figure 1.1: Growth of new blogs by Technorati study (doubling every 6 months).

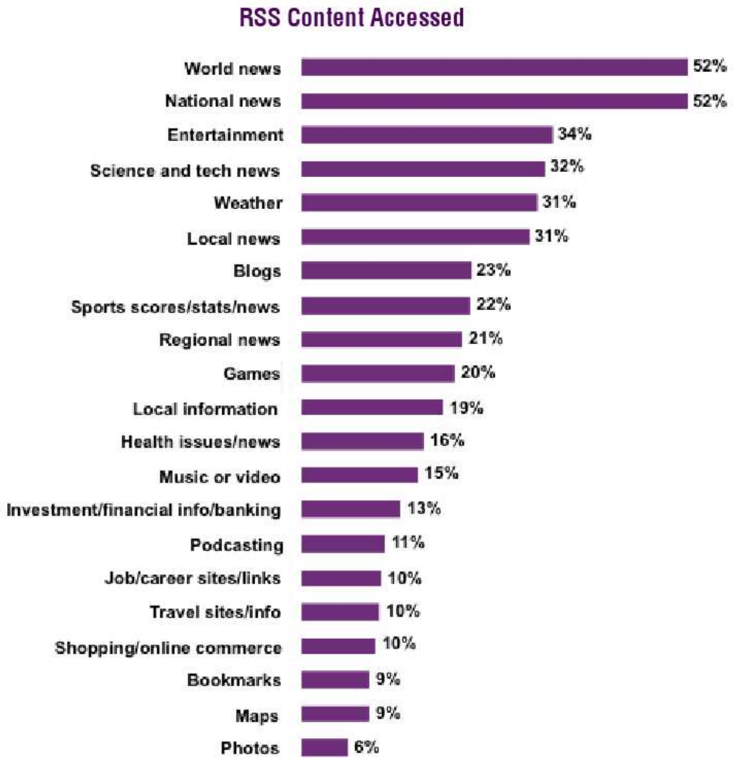
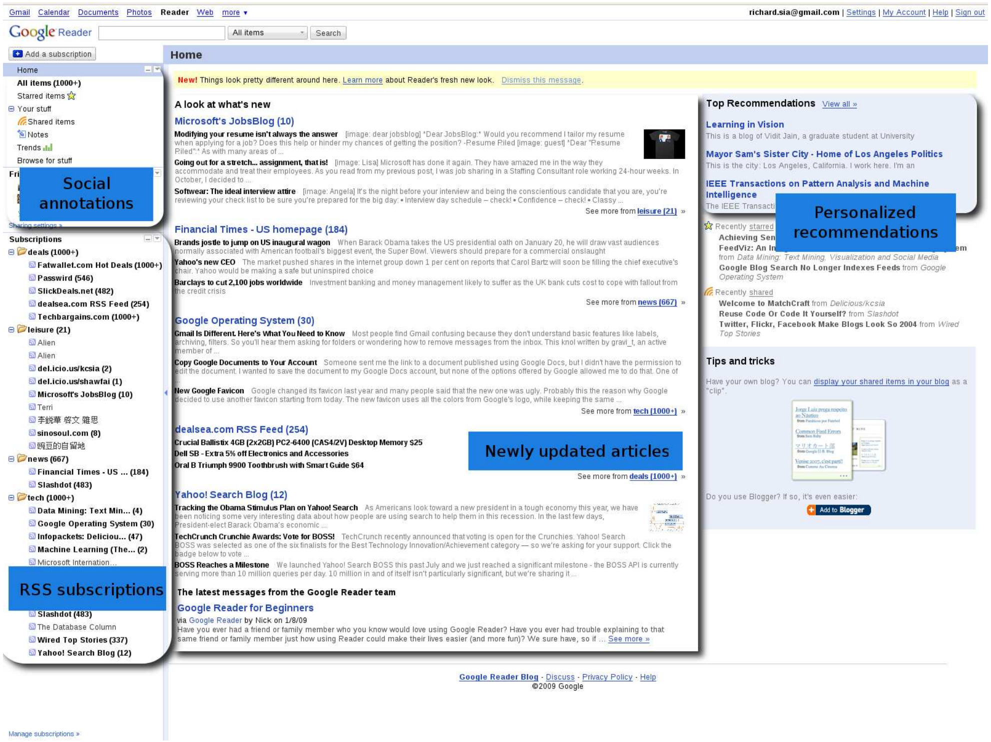


Figure 1.2: Usage of RSS content.

therefore, allows a user to look at new content at a single place, as opposed to at hundreds of distributed sources that she has to visit individually.

A study by Grossnickle *et al.* [GBP05], shown in Figure 1.2, reveals that the majority of content disseminated through RSS feeds are news, blogs, music / video (podcasts), and shopping / online commerce. It suggests that RSS feeds are mainly responsible for delivering everyday online content.

Engineers have built *online RSS readers*, such as Bloglines [Blob] and Google reader [Gooc], to enable ubiquitous access to users. Figure 1.3 shows an example of the Google reader interface. Through such an online RSS reader, users can manage their subscribed RSS feeds (lower left pane) and follow closely any new



Home

All items (1000+)
Starred items
Your stuff
Shared items
Notes
Trends
Browse for stuff

Fri

Sharing settings »

Subscriptions

deals (1000+)
- Fatwallet.com Hot Deals (1000+)
- Passwird (546)
- SlickDeals.net (482)
- dealsea.com RSS Feed (254)
- Techbargains.com (1000+)

leisure (21)
- Alien
- Alien
- del.icio.us/kcsia (2)
- del.icio.us/shawfai (1)
- Microsoft's JobsBlog (10)
- Terri
- 李銳華 啓文 雜思
- sinosoul.com (8)
- 豌豆的自留地

news (667)
- Financial Times - US ... (184)
- Slashdot (483)

tech (1000+)
- Data Mining: Text Min... (4)
- Google Operating System (30)
- Infopackets: Deliciou... (47)
- Machine Learning (The... (2)
- Microsoft Internation...
- Slashdot (483)
- The Database Column
- Wired Top Stories (337)
- Yahoo! Search Blog (12)

Manage subscriptions »

Home

New! Things look pretty different around here. Learn more about Reader's fresh new look. Dismiss this message.

A look at what's new

Microsoft's JobsBlog (10)

Modifying your resume isn't always the answer [image: dear jobsblog] *Dear JobsBlog:* Would you recommend I tailor my resume when applying for a job? Does this help or hinder my chances of getting the position? -Resume Riled [image: guest] *Dear "Resume Riled":* As with many areas of ...

Going out for a stretch... assignment, that is! [image: Lisa] Microsoft has done it again. They have amazed me in the way they accommodate and treat their employees. As you read from my previous post, I was job sharing in a Staffing Consultant role working 24-hour weeks. In October, I decided to ...

Softwear: The ideal interview attire [image: Angela] It's the night before your interview and being the conscientious candidate that you are, you're reviewing your check list to be sure you're prepared for the big day: • Interview day schedule – check! • Confidence – check! • Classy ...

See more from leisure (21) »

Financial Times - US homepage (184)

Brands jostle to jump on US inaugural wagon When Barack Obama takes the US presidential oath on January 20, he will draw vast audiences normally associated with American football's biggest event, the Super Bowl. Viewers should prepare for a commercial onslaught

Yahoo's new CEO The market pushed shares in the internet group down 1 per cent on reports that Carol Bartz will soon be filling the chief executive's chair. Yahoo would be making a safe but uninspired choice

Barclays to cut 2,100 jobs worldwide Investment banking and money management likely to suffer as the UK bank cuts cost to cope with fallout from the credit crisis

See more from news (667) »

Google Operating System (30)

Gmail Is Different. Here's What You Need to Know Most people find Gmail confusing because they don't understand basic features like labels, archiving, filters. So you'll hear them asking for folders or wondering how to remove messages from the inbox. This knol written by gravi_t, an active member of ...

Copy Google Documents to Your Account Someone sent me the link to a document published using Google Docs, but I didn't have the permission to edit the document. I wanted to save the document to my Google Docs account, but none of the options offered by Google allowed me to do that. One of ...

New Google Favicon Google changed its favicon last year and many people said that the new one was ugly. Probably this the reason why Google decided to use another favicon starting from today. The new favicon uses all the colors from Google's logo, while keeping the same ...

See more from tech (1000+) »

dealsea.com RSS Feed (254)

Crucial Ballistix 4GB (2x2GB) PC2-6400 (CAS4/2V) Desktop Memory $25

Dell SB - Extra 5% off Electronics and Accessories

Oral B Triumph 9900 Toothbrush with Smart Guide $64

See more from deals (1000+) »

Yahoo! Search Blog (12)

Tracking the Obama Stimulus Plan on Yahoo! Search As Americans look toward a new president in a tough economy this year, we have been noticing some very interesting data about how people are using search to help them in this recession. In the last few days, President-elect Barack Obama's economic ...

TechCrunch Crunchie Awards: Vote for BOSS! TechCrunch recently announced that voting is open for the Crunchies. Yahoo! Search BOSS was selected as one of the six finalists for the Best Technology Innovation/Achievement category — so we're asking for your support. Click the badge below to vote ...

BOSS Reaches a Milestone We launched Yahoo! Search BOSS this past July and we just reached a significant milestone - the BOSS API is currently serving more than 10 million queries per day. 10 million in and of itself isn't particularly significant, but we're sharing it ...

The latest messages from the Google Reader team

Google Reader for Beginners

via Google Reader by Nick on 1/8/09

Have you ever had a friend or family member who you know would love using Google Reader? Have you ever had trouble explaining to that same friend or family member just how using Reader could make their lives easier (and more fun)? We sure have, so if ... See more »

Top Recommendations View all »

Learning in Vision

This is a blog of Vidit Jain, a graduate student at University

Mayor Sam's Sister City - Home of Los Angeles Politics

This is the city: Los Angeles, California. I work here. I'm an

IEEE Transactions on Pattern Analysis and Machine Intelligence

The IEEE Transacti

Recently starred

Achieving Sen

FeedViz: An In **em** from *Data Mining: Text Mining, Visualization and Social Media*

Google Blog Search No Longer Indexes Feeds from *Google Operating System*

Recently shared

Welcome to MatchCraft from *Delicious/kcsia*

Reuse Code Or Code It Yourself? from *Slashdot*

Twitter, Flickr, Facebook Make Blogs Look So 2004 from *Wired Top Stories*

Tips and tricks

Have your own blog? You can display your shared items in your blog as a "clip".

Do you use Blogger? If so, it's even easier:

Google Reader Blog - Discuss - Privacy Policy - Help

©2009 Google

Figure 1.3: Google reader example.

articles posted. Given the increasing number of RSS feeds a user may have subscribed, it becomes difficult to follow every piece of new articles; therefore, the service provides users an executive-summary style of *what's new and important* (center pane). It ranks articles based on their previous reading patterns or source popularity. Besides, it analyzes the subscribed RSS feed content and recommends interesting topics, often in the form of key phrases and Webpage links, for users to further explore if they are interested (top right pane). In addition, users can also mark certain articles as their "favorites", tagging them with their own keywords for later reference or retrieval and share with friends (upper left pane). With all these functionalities described, the online RSS reader helps users better manage and access online content than using a stand-alone desktop RSS reader program.

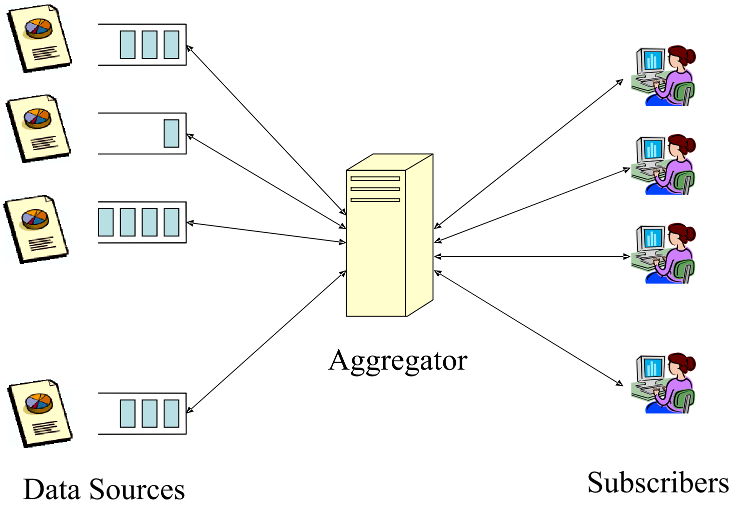


Figure 1.4: An online personalized RSS reader.

1.1 Challenges and opportunities

In this dissertation, I study the challenges and opportunities in building an online personalized RSS reader that provides users with the functionalities mentioned above. As shown in Figure 1.4, such service has to periodically retrieve new content from the list of RSS feeds that are subscribed by its users. It then actively "pushes" the new content to each user or makes it available on a user's next visit, depending on the user's preference. At the same time, it keeps track of all users' profiles, such as their reading patterns and interests, taggings and sharing of articles among friends, in order to provide better user experience through personalization.

While the benefits are enormous, the construction of a large-scale personalized RSS reader raises many interesting engineering challenges. In particular, I investigate the following challenges arising in this context:

1. *Efficient Monitoring*: Due to the large number of RSS feeds that the system

has to monitor and its limited network bandwidth, there can be a significant delay between the publication of new content at the RSS feeds and the retrieval by our system. Since the content published through RSS feeds is often time-sensitive [GBP05] and their readers expect to see new content quickly [AA05], it is of paramount importance to develop an efficient RSS feed monitoring and retrieval policy that minimizes the retrieval delay by the system using the available bandwidth. How should the system schedule the monitoring and the retrieval of RSS feeds to minimize the delay?

2. *Ranking of Articles*: Given that a large number of articles are being generated daily by the set of RSS feeds subscribed by a user, one might find it difficult to read every piece of news. As a way to help users locate relevant and interesting articles, the system has to provide a ranked list of articles based on users' interest. Unfortunately, it is well-known that an average online user is very reluctant to specify her interest explicitly [KB04, SHY04, Hij99], so in order to provide personalized ranking of articles to a user, the system has to *automatically* infer a user's interest based on her past activities. How should the system learn the user's interest from her activities? How to derive a good ranking function of articles based on the learned user interest?

3. *Efficient Computation of Personalized Recommendations*: As another way to improve user experience, the system processes content from all subscribed RSS feeds of a user and analyze what are the most frequently mentioned phrases and Webpages links within the feeds to provide recommendations of interesting topics to users. Unfortunately, computing personalized recommendations for millions of users is prohibitively expensive even for large-scale online service providers. Is there a way to make personalized recommendations for every user without putting an unreasonable strain on

computing resources?

4. *Mining of User Generated Data*: The data, such as taggings of Web resources, that the system collects over time from both the RSS feeds and the interactions of its users can be *mined* to help the users in their other activities. How can we mine this new time-evolving data? What can we learn from this diverse and potentially chaotic data that the system collects?

1.2 Organization of the dissertation

To address the challenges mentioned above, the remainder of this dissertation is organized as follows:

Chapter 2: Monitoring RSS feeds - I start by analyzing the problem faced by RSS aggregation services and also blog search engines that try to provide users a central access point of updated content from a large number of diverse RSS sources. I study the content generation and user-access patterns of RSS feeds and find that they often experience fluctuating yet repeating patterns when investigated at a time scale of hours or less. Such an observation is different from what was previously assumed in the literature, thus, it inspires me to introduce the *periodic inhomogeneous Poisson process* to better model the article-update and user-access process.

Based on the proposed model, I then develop a new content retrieval policy for RSS feeds to deliver "fresh" content to users with minimal *delay* in resource-constrained settings. The proposed policy utilizes both resource allocation (i.e. allocating more retrievals to sources of higher importance, such as those update more frequently and with more subscribers) and retrieval scheduling (i.e. scheduling retrievals closer to updates of articles to reduce the *delay.*) techniques.

Using experiments with real dataset collected from about 10k RSS feeds and user-access patterns from a few volunteers using a Web browser plug-in, I show that the proposed policy significantly reduce the *delay* compared to existing methods. In addition, I investigate a simple change in the RSS protocol that can potentially lead to significant improvement in retrieval delay and estimate its benefit using the real dataset.

Chapter 3: Ranking of articles - In order for a user to keep up with the increasing number of new articles being generated every day, the aggregator often provides a ranked list of articles based on her interest on the front page. This requires the aggregator to learn the user's interest from her previous online activities and devise a good ranking function of articles based on the learned interest.

In Chapter 3, I, together with my collaborators, investigate how we can devise a good ranking function of articles that can *automatically* captures the user's interest based on her activities in the system. In particular, we model a user's interest as a distribution over a set of pre-defined topics and propose a probabilistic framework to model the behavior of reading and clicking of articles in a ranked list. Under this framework, we use both *Exploitation* (showing articles that are of the user's main interest) and *Exploration* (preferring articles from topics that have been less presented to the user so far in order to discover the user's potential interests in those topics) techniques to rank articles among different topics and capture the user's interest quickly through an iterative process.

A rigorous evaluation of the proposed techniques is clearly difficult due to the subjective nature of "good" recommendations, which necessitates an evaluation study with the participation of a large body of unbiased volunteers. Given our limited resources for a large scale human study, we first conduct extensive simulation with synthetic data to explore the key properties of our proposed method and how it compares with a greedy approach that utilizes Exploitation only under simulation. We also conduct a small scale pilot user study with 10 volunteers in technical fields. While our result is not conclusive due to the potential bias in our evaluation group and its small scale, it hints that combining both Exploration and Exploitation has the potential to significantly boost the overall effectiveness of ranking of RSS articles.

Chapter 4: Efficient personal recommendations - Sometimes, online content aggregators may provide an additional functionality to improve user experience. Such a service analyzes the content within all subscribed RSS feeds of a users and recommends popular key phrases and Web links that are being mentioned frequently in those feeds, for her further investigation. In Chapter 4, I explore how we can provide *personalized recommendations* to users based on the set of RSS feeds that they subscribe to. In particular, I address the scalability issues when there are potentially millions of users and millions of sources served by the system.

I start by modeling the personal recommendation computation as matrix multiplications and discover that users often share similar interests. Utilizing this property, I apply the *Non-negative Matrix Factorization* [Hoy04] to cluster users into different interest groups based on their subscription-list similarity. The computation is then broken down into two stages: precomputation of recommendations for user groups and combination of precomputed results for individual users. The computation is further speeded up by an existing top-k computation algorithms, *Threshold Algorithm* [FLN01], to avoid processing unnecessary items.

Using experiments with a dataset collected from an online content aggregator, *Bloglines*, I show that the personalized recommendation approach indeed makes a significant difference in the set of recommended items and it is possible to compute recommendations at a significantly lower cost than by existing methods. In addition, I also discuss the adjustment of parameters to further improve the efficiency of the proposed method.

Chapter 5 Social annotation analysis - Over time, the online RSS aggregator collects a huge amount of user interaction data, often in the form of *social annotations*. The enormous number of social annotations that are being generated by users (e.g., the "tags" used for bookmarking blog articles) involve significant human efforts in matching the best descriptive words with their corresponding Web resources. Such data provide not only rich information to organize Web resources for better retrieval performance but also important clues on vocabulary-usage behavior in the Web information retrieval.

As an example of leveraging this rich body of data to help users, in this chapter, I study the possibility of using the evolving online tagging data to capture the evolutionary characteristics of words, such as their specificity, emergence, and time-sensitivity. To investigate these properties, I apply a state-of-the-art text-mining algorithm, Latent Dirichlet Allocation [BNJ03], to discover the hidden topics among Web resources and their associated keywords. I also draw upon other features extracted from this dataset such as change of keyword membership, diversity of the membership, and change of keyword entropy, etc., then, I apply classifier techniques to draw conclusions on three properties of a word (specificity, emergence, and time-sensitivity) by combining the above mentioned features.

CHAPTER 2

Monitoring RSS feeds

2.1 Introduction

As the popularity of “weblogs” (or blogs) has been growing over time and as many users have been closely following the new postings of their favorite blogs, there has been a dramatic increase in the use of XML data to deliver information over the Web. In particular, personal weblogs, news Websites, and discussion forums are now delivering up-to-date postings to their subscribers using the RSS protocol [RSSa]. To help users access new content in this RSS domain, a number of RSS aggregation services and blog search engines have appeared recently and are gaining popularity [Blob, Bloc, RSSb, Tec, Gooc]. Using these services, a user can either (1) specify the set of RSS sources that she is interested in, so that the user is notified whenever new content appears at the sources (either through email or when the user logs into the service) or (2) conduct a keyword-based search to retrieve all content containing the keyword. Clearly, having a central access point makes it significantly simpler to discover and access new content from a large number of diverse RSS sources.

In this chapter, I investigate an important challenge in building an effective RSS aggregator: How can we minimize the delay between the publication of new content at a source and its appearance at the aggregator and improve the user’s experience by preventing her from missing new content? Note that the aggregation can be done either at a desktop (e.g., RSS-feed readers) or at a central server (e.g., Bloglines [Blob] or Google Reader [Gooc]). In this chapter, I address both the server-based aggregation and a desktop application scenario. This problem is similar to the index refresh problem for Web-search engines [CG00, CG03a, CN02, CLW98, EMT00, PO05, PRC03, WSY02], but two important properties of the information in the RSS domain make this problem unique and interesting:

- The information in the RSS domain is often *time-sensitive*. Most new RSS content is related to current world events, so its value and significance deteriorates rapidly as time passes. An effective RSS aggregator, therefore, has to retrieve new content quickly and make it available to its users close to real time. This requirement is in contrast to general Web search engines where the temporal requirement is not as strict. For example, it is often acceptable to index a new Webpage within, say, a month of its creation for the majority of Webpages.

- For general search engines, users mainly focus on the quality of the *returned pages* and largely ignore (or do not care about) what is not returned [Joa02, LM03]. Based on this observation, researchers have argued for and mainly focused on improving the *quality* of the top k result [PO05], and the page-refresh policies have also been designed to improve the freshness of the top-ranked pages. For RSS feeds, however, many users often have a set of their favorite sources and are particularly interested in reading the new content from these sources. Therefore, users do notice (and complain) if the new content from their favorite sources is missing from the aggregator.

As we will see later, the time-sensitivity of the RSS domain fundamentally changes how we should model the generation of new content in this domain and makes it necessary to design a new content-monitoring policy. In the rest of this chapter, I investigate the problem of how to monitor and retrieve time-sensitive new content effectively from the RSS domain:

- In Section 2.2, I describe a formal framework for this problem. In particular, I propose a *periodic inhomogeneous Poisson process* to model the generation of postings at the RSS feeds and user-access patterns. I also propose to use a *delay* metric and a *miss penalty* metric to evaluate the monitoring policies for RSS feeds.
- In Section 2.3, I develop the optimal ways to retrieve new content from RSS feeds through a careful analysis of the proposed model and metric.
- In Section 2.4, I examine the general characteristics of the RSS feeds based on real RSS-feed data and traces of user-access pattern collected by a browser plug-in. I also evaluate the effectiveness of the retrieval policies using real data. The experiments show that the policy significantly reduces the retrieval delay and new content missed by users compared to the best existing policies.

Note that while my work is primarily motivated by the desire to aggregate the content from the RSS domain, my approach is general and independent of the particular type of the data source (e.g., whether we monitor the content from a general Webpage or from an RSS feed), as will be seen later. As long as the content is time-sensitive and it is important to re-download the content frequently (say, more than once a day), the homogeneous Poisson model becomes less accurate when modeling at a finer time scale, which makes my new approach important.

2.2 Framework

Figure 1.4 illustrates the high-level architecture of an online RSS aggregator, which is a distributed information system that consists of n *data sources*[1], a single *aggregator* and a number of *subscribers*. The data sources constantly generate new pieces of information referred to as new *postings*. I assume a *pull*-based architecture, where the aggregator periodically collects the most recent k postings from each source.[2] A subscriber, in turn, consumes new postings from the aggregator. There does exist another *push*-based architecture where data sources notify ping servers through an XML-RPC protocol whenever there is a new posting. Upon receiving such messages, the aggregator then decides when to retrieve new postings.

Resource constraints

Let's assume that both the aggregator and the sources have limited computational and network resources for the retrieval of new postings. For example, the aggregator may have dedicated T1 lines that allow the aggregator to contact the sources up to one million times per day, or due to the limit of its networking stack, the aggregator may issue up to 500,000 HTTP requests per hour. In this chapter, I model the resource constraint by assuming that the aggregator can contact the sources a total of M times in each period. (The notion of "period" will become clear when I discuss the posting generation model.)

[1] In here, one data source typically corresponds to a single RSS feed, but if multiple RSS feeds can be downloaded through one single HTTP connection to a Web server, they may be grouped together and be considered as one data source.

[2] k is typically in the range of 10–15

Retrieval delay

An effective RSS aggregator has to retrieve new postings from the sources quickly and make them available to its users with minimal delay. The notion of delay can be formalized as follows:

DEFINITION 1 Consider a data source O that generates postings at times $t_1, \ldots, t_k$. t_i is also used to represent the posting itself generated at time t_i unless it causes confusion. The aggregator retrieves new postings from O at times $\tau_1, \ldots, \tau_m$. The delay associated with the posting t_i is defined as

$$D(t_i) = \tau_j - t_i$$

where τ_j is the minimum value with $t_i \leq \tau_j$. The total delay of the postings from source O is defined as

$$D(O) = \sum_{i=1}^{k} D(t_i) = \sum_{i=1}^{k} (\tau_j - t_i) \ \textit{with} \ t_i \in [\tau_{j-1}, \tau_j]. \qquad \square$$

Miss penalty

In some situations where the user-access pattern is known, such as when the aggregation is done at a desktop (e.g. an RSS reader software program) or when the online aggregator records the access pattern of all subscribers of a feed, we may consider another metric that resembles a user's experience when accessing the local copies. Figure 2.1 illustrates a scenario where a data source generates new postings at times $t_i's$, while the aggregator retrieves them at time τ_1, and the user accesses the local copies (as retrieved by the aggregator) at time u_1. Apparently, the user will miss two recently generated postings (t_4 and t_5) at u_1 since the local copies were retrieved before t_4.

Under this scenario, a *miss penalty* metric, which is the number of postings missed by the user when she looks at the local copies, may be used to evaluate

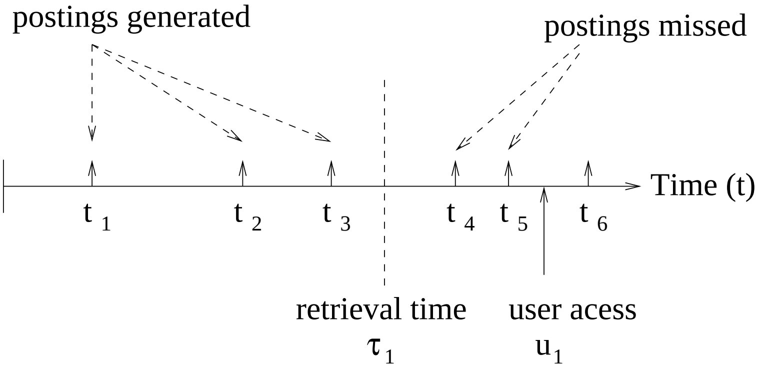


Figure 2.1: Illustration of relationship between number of articles missed, retrieval time and user-access time

the performance of the aggregator. Given a data source O and a user U [3], the metric is defined as follows:

$$M(O,U) = |t_i, \tau_j < t_i \leq u_k| \tag{2.1}$$

where τ_j is the maximum value with $t_i > \tau_j$, and u_k is the minimum value with $t_i < u_k$.

It is also possible to use the metrics that have been widely used in the general search-engine research [CG00, CG03b, CLW98], such as *freshness* and *age*, by modeling the publication of new postings as *modifications* of a data source. For example, the freshness, $F(O;t)$, and age, $A(O;t)$, of a data source O at time instance t can be defined as

$$\begin{aligned} F(O;t) &= \begin{cases} 0 & \textit{if}\ \exists t_i \in [\tau_j, t] \\ 1 & \textit{otherwise} \end{cases} \\ A(O;t) &= \begin{cases} t - t_m & \textit{if}\ \exists t_i \in [\tau_j, t] \\ 0 & \textit{otherwise} \end{cases} \end{aligned}$$

[3] A user can be an individual or a group of users who have subscribed to the same data source.

where τ_j is the most recent retrieval time and t_m is the minimum of all t_i's within $[\tau_j, t]$.

For illustration, Figure 2.2 (a), (b), and (c) show an example evolution of delay, freshness, and age respectively. The data source generates five postings at $t_1, \ldots, t_5$ (marked by dashed lines). Two retrievals are scheduled by the aggregator at τ_1 and τ_2 (marked by solid lines). The vertical axes represent the delay, freshness, and age associated with the data source. Note that after the generation of t_2, the delay metric increases twice as rapidly as before because two new postings, t_1 and t_2, are pending at the source. In contrast, the age metric does not take into account those two pending postings and still increases at the same constant rate as before. Thus, the delay metric can be considered as an improved version of the age metric that takes into account multiple postings pending at a source, which is more appropriate in the context of RSS feeds.

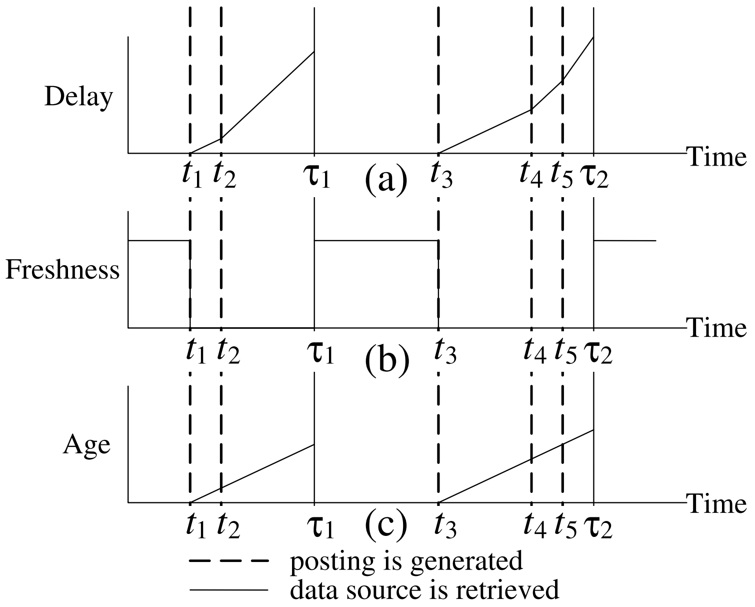

Delay
Time
t_1 t_2 τ_1 (a) t_3 t_4 t_5 τ_2
Freshness
Time
t_1 t_2 τ_1 (b) t_3 t_4 t_5 τ_2
Age
Time
t_1 t_2 τ_1 (c) t_3 t_4 t_5 τ_2
posting is generated
data source is retrieved

Figure 2.2: Illustration of the delay, freshness, and age metrics

When multiple sources generate new postings, it may be more important to minimize the delay from one source than others. For example, if a source has more subscribers than others, it may be more beneficial to minimize the delay for this source. This difference in importance is captured in the following weighted definition:

Definition 2 Let's assume each source O_i is associated with weight w_i. Then the total weighted delay observed by the aggregator, $D(A)$, is defined as

$$D(A) = \sum_{i=1}^{n} w_i \, D(O_i) \qquad \square$$

Delay minimization problem

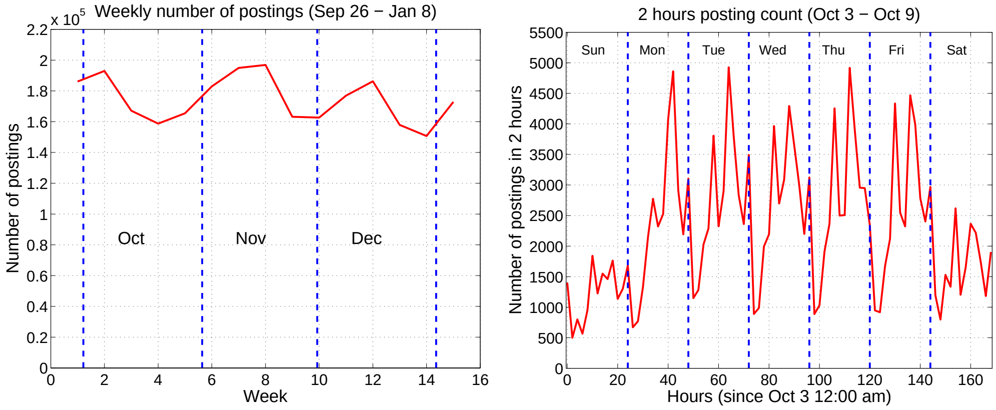
When t_{ij} is used to represent the jth posting generation time at O_i and τ_{ij} to refer to the time of the jth retrieval from O_i by the aggregator, the problem of delay minimization is formalized as follows:

Problem 1 Given the posting generation times t_{ij}'s, find the retrieval times τ_{ij}'s that minimize the total delay $D(A) = \sum_{i=1}^{n} w_i \, D(O_i)$ under the constraint that the aggregator can schedule a total of M retrievals. $\square$

2.2.1 Posting generation model and user-access pattern

In practice, the aggregator does not know the future posting generation times t_{ij}'s. Therefore, to solve the delay minimization problem, the aggregator has to *guess* the future posting times based on the *past* posting pattern of each source.

In the context of general Web search engines, researchers have proposed that a *homogeneous* Poisson process with a rate λ is a good model to be used [CG00, CG03b]. Roughly, a homogeneous Poisson process is a stateless and time-independent random process, where new postings always appear at a *constant* rate λ regardless of the time [TK98]. A number of studies [CG00, CG03b] show that this model is



(a) Weekly posting rate

(b) Hourly posting rate

Figure 2.3: Posting rate at different resolution.

appropriate especially when the time granularity is longer than one month. For example, Figure 2.3(a) shows the total number of postings appearing in roughly 10,000 RSS feeds being monitored. (More details of this dataset are described in the experiment section). The horizontal axis is the time, and the vertical axis shows the number of postings appearing in each week of the monitoring period. While there are small fluctuations, the total number of new postings in each week is reasonably stable at roughly 180,000 postings, which matches with the homogeneous Poisson assumption. Formally, this assumption can be stated as $\lambda(t) = \lambda$, where the posting generation rate at time t, $\lambda(t)$, is constant and independent of time t. Based on this homogeneous model, researchers have derived the optimal re-download algorithms for Web crawlers [CG03b, CLW98].

Unfortunately, when the time granularity is much shorter than one month, there exists strong evidence that the homogeneous Poisson model is no longer adequate [BC00, GE01, GGL04]. For example, Figure 2.3(b) shows the total number of postings appearing in the same RSS feeds when we count the number at a granularity of two hours. From the figure, it is clear that at this time granularity, the time-independence property of the homogeneous Poisson model does not hold. The posting rate goes through wide fluctuation depending on the time of the day and the day of the week. The graph also shows a certain level of periodicity in the posting rates. During the day, there is a significantly higher number of postings than at night. Similarly, there are more activities during the weekdays than on weekends. Not only the aggregated posting rate shows such a pattern, individual RSS feeds also exhibit daily and weekly fluctuations but with different patterns as shown in Figure 2.12. Based on this observation, I propose using an *inhomogeneous* Poisson model, where the posting rate $\lambda(t)$ changes over time. Depending on whether similar patterns of $\lambda(t)$ values are repeated over time, this model can be further classified into one of the following:

1. *Periodic inhomogeneous Poisson model*: The same $\lambda(t)$ values are repeated over time with a period of T. That is, $\lambda(t) = \lambda(t - nT)$ for $n = 1, 2, \ldots$. This model may be a good approximation when similar rate patterns are repeated over time, such as the burst of activities during the day followed by a period of inactivity at night.

2. *Non-periodic inhomogeneous Poisson model*: This is the most general model where no assumption is made about the periodicity in the changes of $\lambda(t)$. That is, there exists no T that satisfies $\lambda(t) = \lambda(t - nT)$.

Not surprisingly, user-access pattern also follows a similar trend. Figure 2.4 shows a particular user's Webpage access activities during a 2-week time period collected by a browser plug-in. Each bar in the figure represents the number of Webpages accessed by the user within the corresponding 2-hour period. From the figure, certain periodicity is observed: the user makes significantly more accesses

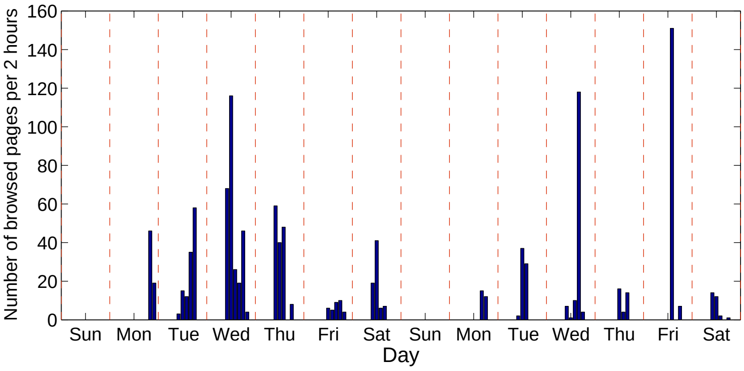


Figure 2.4: A sample 2-weeks' user-access pattern.

during the day than in the night and during the weekdays than on the weekends. To illustrate this periodic fluctuation more clearly, Figure 2.19 shows the average number of Webpages accessed per hour aggregated from 9 different users within one day when we overlap their access history over 2 weeks. The graphs show fluctuations that seem to correspond to users' work schedule (e.g. lunch and dinner breaks, sleeping habits, etc.). Such observations suggest that a periodic inhomogeneous Poisson process [TK98] may be a good approximation for such a recurring and fluctuating pattern.

Given the periodicity that is observed in the RSS posting pattern and the user-access pattern, I mainly use the periodic inhomogeneous Poisson model in the rest of this chapter.

2.2.2 Expected retrieval delay

Since the aggregator does not know the exact times at which new postings are generated, it can only estimate the *expected* delay based on the posting generation model of a source. In general, the expected delay can be computed as follows under the general inhomogeneous Poisson model:

Lemma 1 *For a data source O with the rate $\lambda(t)$, the total expected delay for the postings generated within $[\tau_{j-1}, \tau_j]$ is as follows:*

$$\int_{\tau_{j-1}}^{\tau_j} \lambda(t)(\tau_j - t)dt. \qquad \square$$

Proof During a small time interval dt at time t, $\lambda(t)dt$ postings are generated. Since these postings are retrieved at time τ_j, their associated delays are $\tau_j - t$. Therefore, the total delay of the postings generated between τ_{j-1} and τ_j is $\int_{\tau_{j-1}}^{\tau_j} \lambda(t)(\tau_j - t)dt$. ■

For the simpler homogeneous Poisson model, the above formula is simplified to the following formula.

Corollary 1 *When the posting rate remains constant at λ within the time period $[\tau_{j-1}, \tau_j]$, the total expected delay for postings generated within this period is*

$$\frac{\lambda(\tau_j - \tau_{j-1})^2}{2}. \qquad \square$$

2.2.3 Expected miss penalty

Likewise, the aggregator has to estimate the *expected miss penalty* based on previously learned user-access patterns. The following lemma shows how to compute the expected miss penalty under the proposed model:

Lemma 2 *For a data source O with posting rate $\lambda(t)$ and a user U with access rate $u(t)$, assume the retrievals are scheduled at time τ_{j-1} and τ_j. Then the expected penalty experienced by the user during the time period within $[\tau_{j-1}, \tau_j]$ is as follows:*

$$\int_{\tau_{j-1}}^{\tau_j} u(t)(\int_{\tau_{j-1}}^{t} \lambda(x)dx)dt \qquad \square$$

Proof During a small time interval dt at time t, there are $u(t)dt$ number of user accesses. Each access will miss $\int_{\tau_{j-1}}^{t} \lambda(x)dx$ number of posts. Therefore, the total expected *miss penalty* experienced by the user between τ_{j-1} and τ_j is $\int_{\tau_{j-1}}^{\tau_j} u(t)(\int_{\tau_{j-1}}^{t} \lambda(x)dx)dt$. ■

The expected delay and miss penalty computed above will be used in the next section when investigating the optimal retrieval policy used by the aggregator. $\Lambda(t)$ and $U(t)$ are used to denote the integrals $\int_0^t \lambda(x)dx$ and $\int_0^t u(x)dx$ respectively from now on.

2.3 Retrieval policy

I now study how the aggregator should schedule the M retrieval points τ_{ij}'s to minimize the total expected delay. This scheduling problem is broken down in two steps:

- *Resource allocation*: Given n data sources and a total of M retrievals per period T, the aggregator first decides *how many times* it will contact individual source O_i. This decision should be made based on how frequently new postings appear in each source and how important each source is.
- *Retrieval scheduling*: After the aggregator decides how many times it will contact O_i per T, it decides exactly *at what times* it will contact O_i. For example, if the aggregator has decided to contact O_1 twice a day, it may either schedule the two retrieval points at uniform intervals (say, one at midnight and one at noon) or it may schedule both retrievals during the day when there are likely to be more new postings.

In Section 2.3.1, I start with the resource-allocation problem, then followed by a study of the retrieval-scheduling problem in Section 2.3.2. As far as I know, my work is the first study to develop optimal solutions for the retrieval-scheduling problem for Web sources, while similar resource-allocation problems have been studied before (e.g., [CG00, CG03a, CLW98]), albeit under a different metric. Finally, in Section 2.4.4, I go over techniques to obtain an accurate estimate of posting rates and patterns from past history.

2.3.1 Resource-allocation policy

This section investigates how to allocate the M retrievals among the data sources to minimize the total expected delay. For this task, the simple homogeneous Poisson process model is used because the resource allocation is done based on the *average posting generation rate* and the *weight of each source*, both of which are adequately captured by the homogeneous Poisson model. The more complex inhomogeneous model will be used later when we consider the retrieval-scheduling problem.

The main result for this resource-allocation problem is summarized in the following theorem, which shows that the optimal allocation of resources to a source O_i should be proportional to the square root of the product of its posting rate λ_i and its importance w_i.

Theorem 1 *Consider data sources $O_1, \ldots, O_n$, where O_i has the posting rate λ_i and the importance weight w_i. The aggregator performs a total of M retrievals per each period T.*

Under this scenario, the weighted total delay of postings, $D(A) = \sum_{i=1}^{n} w_i D(O_i)$, becomes minimum when the source O_i is contacted at a frequency proportional to $\sqrt{w_i \lambda_i}$. That is, m_i, the optimal number of retrievals per each period for O_i, is *given by*

$$m_i = k\sqrt{w_i \lambda_i} \tag{2.2}$$

where k is a constant that satisfies $\sum_{i=1}^{n} k\sqrt{w_i\lambda_i} = M$. □

Proof Let's consider the data source O_i that is retrieved m_i times per day. Under the homogeneous Poisson model, it can be shown that $D(O_i)$, the total delay of postings from O_i, is minimum when the retrievals are scheduled at the uniform interval.[4] In this case, $D(O_i) = \frac{\lambda_i T^2}{2m_i}$, and the total weighted delay, $D(A)$, is

$$D(A) = \sum_{i=1}^{n} \frac{\lambda_i w_i T^2}{2m_i}.$$

$D(A)$ can be minimized by using the Lagrange multiplier method.

$$\frac{\partial D(A)}{\partial m_i} = -\frac{\lambda_i w_i T^2}{2m_i^2} = -\mu.$$

If we rearrange the above equation, we get

$$m_i = \sqrt{\lambda_i w_i T^2 / 2\mu} = k\sqrt{\lambda_i w_i}.$$ ■

As we can see from the solution, the optimal resource allocation can be computed simply by multiplying the posting rate of each source with k, which can be computed from w_i's and λ_i's. Therefore, the complexity of computing the optimal resource-allocation policy is linear with the number of data sources.

2.3.2 Retrieval-scheduling policy

I have just discussed how to allocate resources to data sources based on their weights and average posting rates. Assuming that postings are retrieved m times

[4] This proof follows from a special case of the Cauchy's inequality stating that the sum of squares is always less than the square of sums and that equality holds when all numbers are equal.

from the source O, I now discuss exactly at what times we should schedule the m retrievals. Clearly, this decision should be based on what time of the day the source is expected to generate the largest number of postings and what time the user is accessing local copies, so I now use the periodic inhomogeneous Poisson model to capture the daily fluctuation in the posting generation rate and user-access rate.

To make the discussion easy to follow, I start with a simple case when only one retrieval is allocated per period in Section 2.3.2.1. The analysis is then extended to a more general case in Section 2.3.2.2.

2.3.2.1 Single retrieval per period

Consider a data source O at the periodic posting rate $\lambda(t) = \lambda(t - nT)$. The postings from O are retrieved only once in each period T. The following theorem shows that the best retrieval time is when the instantaneous posting rate $\lambda(t)$ equals the average posting rate over the period T.

THEOREM 2 *A single retrieval is scheduled at time τ for a data source with the posting rate $\lambda(t)$ of period T. Then, when the total delay from the source $D(O)$ is minimized, τ satisfies the following condition:*

$$\lambda(\tau) = \frac{\Lambda(T)}{T} \qquad \left(and \ \frac{d\lambda(\tau)}{d\tau} < 0 \right). \tag{2.3}$$ □

PROOF Without loss of generality, only the postings generated within a single interval $[0, T]$ are being considered. The notation $D(\tau)$ is used to represent the delay when the retrieval is scheduled at τ. The postings generated between $[0, \tau]$ are retrieved at τ, so their delay is $\int_0^\tau \lambda(t)(\tau - t)dt$. The postings generated between $[\tau, T]$ are retrieved in the next interval at time $T + \tau$, so their delay is

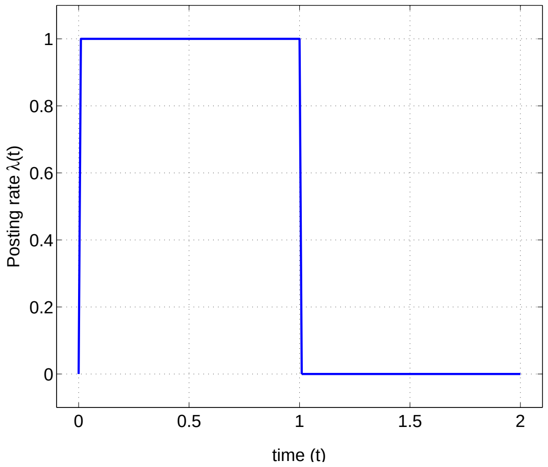


Figure 2.5: A data source going through periods of high activity and low activity.

$\int_{\tau}^{T} \lambda(t)(T+\tau-t)dt$. Therefore,

$$\begin{aligned} D(\tau) &= \int_0^{\tau} \lambda(t)(\tau-t)dt + \int_{\tau}^{T} \lambda(t)(T+\tau-t)dt \\ &= T\int_{\tau}^{T} \lambda(t)dt + \int_0^{T} \lambda(t)(\tau-t)dt. \end{aligned}$$

$D(\tau)$ is minimum when

$$\frac{dD(\tau)}{d\tau} = -T\,\lambda(\tau) + \Lambda(T) = 0$$

and $\frac{d^2D(\tau)}{d\tau^2} = -T\,\frac{d\lambda(\tau)}{d\tau} > 0$. After rearranging the expressions, we get Equation 2.3. ■

The implication of the theorem is illustrated using a simple example.

Example 1 Figure 2.5 shows a data source that goes through a period of high activity, $\lambda(t) = 1$, during $t \in [0,1]$ and a period of low activity, $\lambda(t) = 0$, during $t \in [1,2]$. The same pattern is repeated after $t = 2$. Its postings are retrieved once in each period.

According to Theorem 2, the retrieval should be scheduled at $t = 1$ when the $\lambda(t)$ changes from 1 to 0 and takes the average value $\lambda(t) = 0.5$. This result matches the intuition that the retrieval should be scheduled right after a period of high activity. The expected total delay in this case is $\frac{1}{2}$. Compared to the worst case when the retrieval is scheduled right before a period of high activity (i.e., $\tau = 0$), we get a factor of 3 improvement. Compared to the average case, we get a factor of 2 improvement. □

When a user U, or a group of users, with a periodic access rate $u(t)$ having the same periodicity T as the periodic posting rate $\lambda(t)$ of a data source O is considered, the following theorem suggests that the best retrieval time is when the instantaneous posting rate $\lambda(t)$ is proportional to the instantaneous user-access rate $u(t)$ according to a value given by $\frac{U(T)}{\Lambda(T)}$.

Theorem 3 *When a single retrieval is scheduled at time τ, the expected penalty $M(O,U)$ is minimized when τ satisfies the following conditions:*

$$\frac{u(\tau)}{\lambda(\tau)} = \frac{U(T)}{\Lambda(T)} \qquad \left(and\ \frac{u'(\tau)}{\lambda'(\tau)} > -\frac{U(T)}{\Lambda(T)} \right) \tag{2.4}$$ □

Proof Without loss of generality, only the user accesses within a single interval $[0, T]$ are being considered. The notation $M(\tau)$ represents the expected number of postings missed by the user when the retrieval is scheduled at τ, where $0 \leq \tau \leq T$. Between $[0, \tau]$, the user will miss the postings generated between $[\tau - T, t]$; between $[\tau, T]$, she will miss the postings generated between $[\tau, t]$, where t is the exact time she accesses the local copies. Therefore, the *expected miss penalty* when we schedule one retrieval at time τ is

$$\begin{aligned} M(\tau) &= \int_0^\tau u(t)(\int_\tau^T \lambda(x)dx + \int_0^t \lambda(x)dx)dt \\ &\quad + \int_\tau^T u(t)(\int_\tau^t \lambda(x)dx)dt \end{aligned}$$

$$= \int_0^T u(t)\Lambda(t)dt - \Lambda(\tau)U(T) + \Lambda(T)U(\tau).$$

$M(\tau)$ is minimum when

$$\frac{dM(\tau)}{d\tau} = \Lambda(T)u(\tau) - U(T)\lambda(\tau) = 0$$

and $\frac{d^2M(\tau)}{d\tau^2} > 0$. After rearranging the expressions, we get Equation 2.4. ■

The implication of this theorem is illustrated using a simple example.

Example 2 Figure 2.6 shows a data source (blue solid line) that undergoes a period of high posting activity between $t = [0, 1]$ and a period of low posting activity between $t = [1, 2]$. Similarly, the user (red dashed line) also undergoes a fluctuation in her access pattern, but in a reverse way from the data source. According to the theorem, the best retrieval should be scheduled at $t = 1$, where $u(1) = \lambda(1)$ and $\frac{u'(1)}{\lambda'(1)} > -1$. This solution matches the intuition that the crawler should retrieve right after a large number of new postings are generated and before the user accesses the local copies extensively in order to prevent users from missing too many postings. □

2.3.2.2 Multiple retrievals per period

Now, I generalize the scenario and consider the case when multiple retrievals are scheduled within one period.

Theorem 4 *Suppose m retrievals are scheduled at time $\tau_1, \ldots, \tau_m$ for a data source with the posting rate $\lambda(t)$ and periodicity T. When the total delay is minimized, the τ_j's satisfy the following equation:*

$$\lambda(\tau_j)(\tau_{j+1} - \tau_j) = \int_{\tau_{j-1}}^{\tau_j} \lambda(t)dt, \qquad (2.5)$$

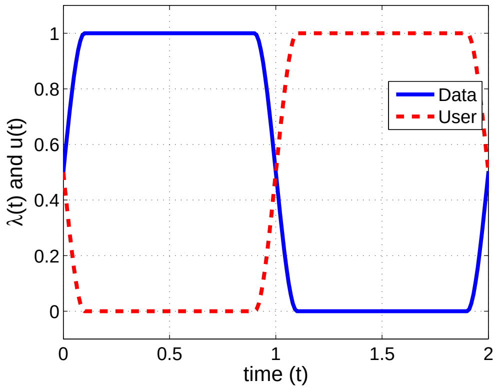


Figure 2.6: Example of the single optimal retrieval point.

where $\tau_{m+1} = T + \tau_1$ *(the first retrieval point in the next interval) and* $\tau_0 = \tau_m - T$ *(the last retrieval point in the previous interval).* □

Proof Without loss of generality, the expected total delay in postings generated between τ_1 and $T + \tau_1$ is considered:

$$\begin{aligned} D(O) &= \sum_{i=1}^{m} \int_{\tau_i}^{\tau_{i+1}} \lambda(t)(\tau_{i+1} - t)dt \\ &= \sum_{i=1}^{m} \left(\tau_{i+1} \int_{\tau_i}^{\tau_{i+1}} \lambda(t)dt \right) - \int_{\tau_1}^{T+\tau_1} \lambda(t)tdt \\ &= \sum_{i=1}^{m} \left(\tau_{i+1} \int_{\tau_i}^{\tau_{i+1}} \lambda(t)dt \right) - \int_{0}^{T} \lambda(t)tdt. \end{aligned}$$

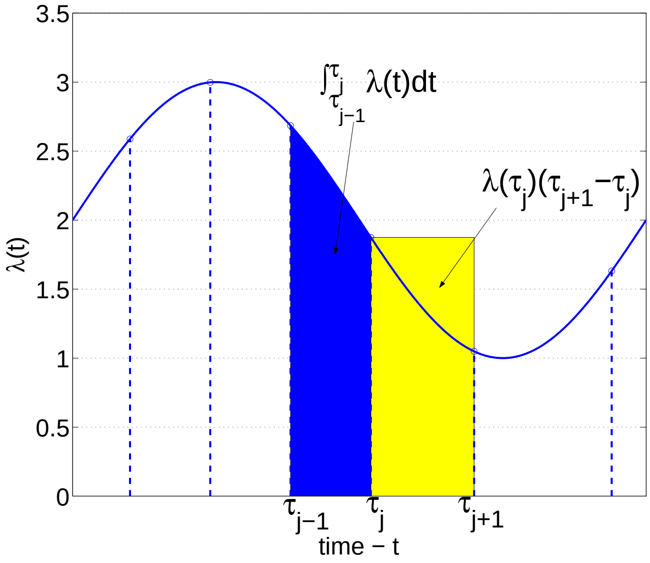
Then $D(O)$ is minimum when $\frac{\partial D}{\partial \tau_j}$ for every τ_j:

$$\frac{\partial D}{\partial \tau_j} = \int_{\tau_{j-1}}^{\tau_j} \lambda(t)dt + \tau_j \lambda(\tau_j) - \tau_{j+1}\lambda(\tau_j) = 0.$$

By rearranging the above expression, we get Equation 2.5. ■

The graphical meaning of the theorem is illustrated using an example.

Example 3 Figure 2.7 shows a data source with the posting rate $\lambda(t) = 2 + 2\sin(2\pi t)$. Postings are retrieved from the source 6 times in one period. Assume



□

Figure 2.7: The optimal schedule for 6 retrievals per period for data source with posting rate $\lambda(t) = 2 + 2\sin(2\pi t)$.

that we have decided up to the j^{th} retrieval point and need to determine the $(j+1)^{th}$ point. Note that the right-hand side of Equation 2.5 corresponds to the dark-shaded area in Figure 2.7. The left-hand side of the equation corresponds to the light-shaded area of Figure 2.7. The theorem states that the total delay is minimized when τ_{j+1} is selected such that the two areas are the same.

When the user-access pattern is also considered and the *expected miss penalty* is being optimized, the following theorem states the optimal condition of scheduling retrievals.

Theorem 5 *When m retrievals at time $\tau_1, \ldots, \tau_m$ are scheduled for a data source with posting rate $\lambda(t)$ based on a user-access rate $u(t)$, where both have periodicity T, the expected penalty is minimized when all $\tau_{j's}$ satisfy the following equation:*

$$\frac{u(\tau_i)}{\lambda(\tau_i)} = \frac{\int_{\tau_i}^{\tau_{i+1}} u(t)dt}{\int_{\tau_{i-1}}^{\tau_i} \lambda(t)dt} \tag{2.6}$$

where $\tau_{m+1} = T + \tau_1$ *(the first retrieval point in the next interval) and* $\tau_0 = \tau_m - T$ *(the last retrieval point in the previous interval).* □

Proof Without loss of generality, the expected penalty of a user when she accesses the content between τ_1 and $T + \tau_1$ is considered:

$$\begin{aligned} M(O,U) &= \sum_{i=1}^{m} \int_{\tau_i}^{\tau_{i+1}} u(t)(\int_{\tau_i}^{t} \lambda(x)dx)dt \\ &= \sum_{i=1}^{n} [(\int_{\tau_i}^{\tau_{i+1}} u(t)\Lambda(t)dt) - \Lambda(\tau_i)(U(\tau_{i+1}) - U(\tau_i))]. \end{aligned}$$

The necessary condition for $M(O,U)$ to be minimum is when $\frac{dM(O,U)}{d\tau_i} = 0$ for every τ_i. By rearranging the terms, we get Equation 2.6. ■

The above theorem is graphically illustrated using an example.

Example 4 Figure 2.8 shows a data source (blue solid line) with the posting rate $\lambda(t) = 2 + sin(2\pi t)$ and a user-access rate (red dashed line) of $u(t) = 2 + cos(2\pi t)$. Postings are retrieved from the source six times in one period. Assume that we have decided up to the i^{th} retrieval point and need to determine the $(i+1)^{th}$ point. Note that the upper part of the right-hand side of Equation 2.6 is equivalent to the dark-shaded area in Figure 2.8, while the lower part of the right-hand side of Equation 2.6 is equivalent to the light-shaded area of Figure 2.8. The theorem states that the expected penalty is minimized when τ_{i+1} is selected such that the two areas are in proportion to $\frac{u(\tau_i)}{\lambda(\tau_i)}$, which is the ratio of the instantaneous user-access rate to the posting rate at time τ_i. □

2.3.3 Computation of schedule

The above theorems only provide analytical solutions to the optimal conditions when $\lambda(t)$ and $u(t)$ are known. In practice, we may need to learn the two patterns

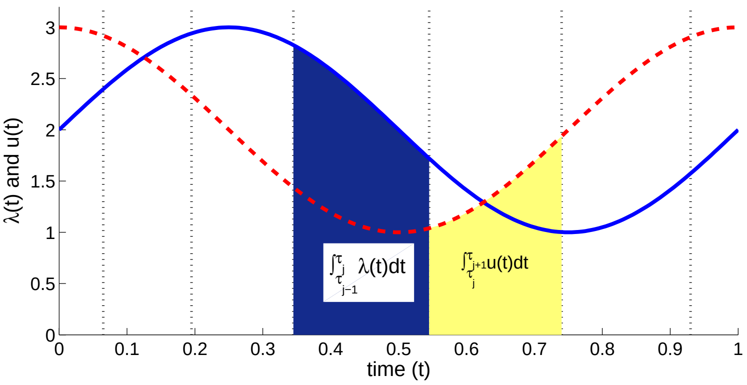


Figure 2.8: The optimal schedule for 6 retrievals per interval.

and discretize the continuous time domain and schedule retrievals at discrete time points. One naive solution to find the optimal schedule is to enumerate all possible retrieval schedules and find the best one with the lowest *expected delay* or *expected miss penalty*. In this section, I describe how to use the conditions derived previously to compute the optimal schedules efficiently.

Suppose we have discretized one day period into 1440 slots (assuming the resolution of retrieval time is set to be one minute). For the single retrieval case, one can use the method of bisection to find the slot that satisfies Equation 2.3 or 2.4, which may converge to the solution faster than searching the slot linearly. For multiple retrievals per interval case, how the condition is utilized to compute the solution is illustrated as follows: Suppose we have picked the first two retrieval points; we keep applying Equation 2.5 or 2.6 iteratively to determine the successive retrieval points. Once we have determined all retrieval points using the conditions, we then compute the *expected delay* or *expected miss penalty* of this particular schedule. This procedure is then repeated for all possible choices of the first two retrieval points and choose the schedule with the lowest outcome; the pseudo-code for this computation is illustrated in Algorithm 1. This computation basically applies the necessary condition in Theorem 4 and 5 to prune out unnecessary search space, as compared to a method that enumerates all possible schedules and searches.

To learn the functions, $\lambda(t)$ and $u(t)$, from the past posting history and the user-access history, we may count the number of postings and the number of user accesses per *hour* and represent them as a 24-bin histogram. The 24-bin histogram is then smoothed out by linear piecewise interpolation (or possibly by some other smoothing functions) to obtain a discretized version of $\lambda(t)$ and $u(t)$. In the following experiments, one hour is chosen as the bin size for the estimation of $\lambda(t)$ and $u(t)$ because when the bin size is much smaller (say, a minute), the histogram may contain many spikes, while when the bin size is much larger (say, 2 or 3 hours), the histogram may not capture the fluctuation precisely, and both extremes will result in poor prediction accuracy.

2.4 Experiments

In this section, I show some statistics of the collected RSS-feeds data, user-access patterns, and the result from the performance evaluation of the proposed retrieval policies.

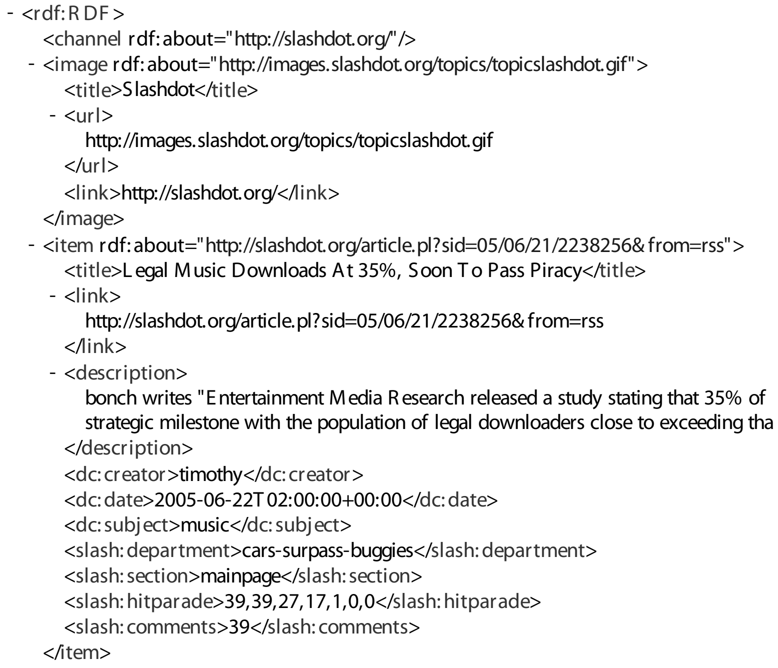
2.4.1 Description of RSS dataset

RSS feeds are essentially XML documents published by Web sites, news agents, or bloggers to ease syndication of their Web site's contents to subscribers. Figure 2.9 shows a typical RSS feed. It contains different postings in the ⟨**item**⟩ tag and summaries in the ⟨**description**⟩ tag. Each posting is associated with a timestamp ⟨**dc:date**⟩, stating when it was generated. The postings are arranged in the

Algorithm 1 optimal schedule algorithm

```
smooth the input histogram with more number of bins.
for all possible position of $\tau_1$
  for all possible positions of $\tau_2$
    while desired number of retrievals not reached
      apply Eq. 2.5 or 2.6 to determine $\tau_{j+1}$
      if $\tau_{j+1}$ exceed one period then
        break
      end if
    end while
    if $\tau_{m-1}, \tau_m, \tau_1, \tau_2$ satisfy Eq. 2.5 or 2.6 then
      compute expected delay or miss penalty of the schedule
    else
      continue
    end if
  end for
end for
select the schedule with smallest expected delay or miss penalty
```

reverse chronological order where new postings are prepended in the front and old postings are pushed downwards and removed. For the majority of current implementations, an RSS feed contains the most recent 10 or 15 postings. New postings are added to the feed at any time without notifying their subscribers; thus, the subscribers have to poll the RSS feeds regularly and check for updates. The set of data used comes from a list of 12K RSS feeds listed in syndic8 [Syn] that includes time of posting within the RSS. They were downloaded 4 times a day between September 2004 and December 2004. Out of the 12K feeds, 9,634 feeds (about 80%) have at least one posting within this monitoring period. The following experiments will all focus on this subset of 9,634 RSS feeds. The range of the topics covered by this set of RSS feeds is quite diverse and the feeds come from about five thousand different domains. Table 2.1 shows some of the frequent domains where these RSS feeds originate from.



```
- <rdf:RDF>
    <channel rdf:about="http://slashdot.org/"/>
  - <image rdf:about="http://images.slashdot.org/topics/topicslashdot.gif">
      <title>Slashdot</title>
    - <url>
        http://images.slashdot.org/topics/topicslashdot.gif
      </url>
      <link>http://slashdot.org/</link>
    </image>
  - <item rdf:about="http://slashdot.org/article.pl?sid=05/06/21/2238256&from=rss">
      <title>Legal Music Downloads At 35%, Soon To Pass Piracy</title>
    - <link>
        http://slashdot.org/article.pl?sid=05/06/21/2238256&from=rss
      </link>
    - <description>
        bonch writes "Entertainment Media Research released a study stating that 35% of
        strategic milestone with the population of legal downloaders close to exceeding tha
      </description>
      <dc:creator>timothy</dc:creator>
      <dc:date>2005-06-22T02:00:00+00:00</dc:date>
      <dc:subject>music</dc:subject>
      <slash:department>cars-surpass-buggies</slash:department>
      <slash:section>mainpage</slash:section>
      <slash:hitparade>39,39,27,17,1,0,0</slash:hitparade>
      <slash:comments>39</slash:comments>
    </item>
```

Figure 2.9: A sample RSS feed

Count	Domain
1133	`scotsman.com`
209	`www.rss-job-feeds.org`
154	`newsroom.cisco.com`
138	`www.employmentspot.com`
118	`blogs.msdn.com`
109	`radio.weblogs.com`
88	`feedster.com`
83	`www.computerworld.com`
79	`www.sportnetwork.net`
67	`abclocal.go.com`

Table 2.1: Top 10 domains hosting the RSS feeds in the dataset.

Figure 2.10 shows the distribution of posting rates among the 9,634 RSS feeds, with the x-axis being the number of postings generated within three months and the y-axis being the number of feeds at the given rate. Both axes are shown in log scale. Within the 3 months, 3,116 feeds generated one or more postings per day on average. The distribution roughly follows a straight line in the log-log scale plot, which suggests that it follows a power-law distribution.[5]

2.4.2 Evaluation of policy under delay metric

In this section, I evaluate the potential improvement of the proposed policy under *delay* metric by comparing it against the best policies in the literature. In particular, I compare the *total weighted delay* $D(A)$ (Definition 2 in Section 2.2) achieved by the proposed policy against that of the age-based optimal crawling

[5] A curve fit of the data indicates that the best matching power-law curve is $y = ax^b$, with $a \simeq 376$ and $b \simeq -0.78$.

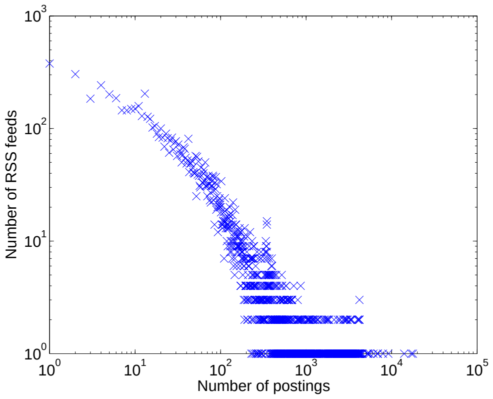


Figure 2.10: Distribution of posting rate of 9,634 RSS feeds.

policy in [CG03a].[6] Since both policies have to know the average posting rate of each source, the rates from the first 2-week data are first computed and then being used to simulate the policies on the remaining 11-week data.[7] Equal weights are assigned to all sources instead of weighting by the number of subscribers because we want to evaluate the improvement from the accurate modeling of the posting generation at the sources, which is the main focus of this chapter. For my proposed policy, both resource-allocation and retrieval-scheduling policies and employed. Note that the *delay* metric measures the time between the publication at the source and the download by the aggregator. Therefore, this set of experiments measures the effectiveness of the optimization based on the data posting pattern alone. The effectiveness of the user-access pattern based optimization will be investigated later when I report results under the *miss penalty* metric.

The results from these experiments are shown in Figure 2.11. The horizontal axis represents the resource constraint for a given experiment. More precisely, it

[6] Reference [CG03a] describes two policies, one for the freshness metric and the other for the age metric. Since the result from the age-based policy outperforms the freshness-based policy by several orders of magnitude, only the age-based policy in the comparison is shown.

[7] The choice of the two-week estimation window is explained later in Section 2.4.4.

shows the *average retrieval interval* per source (i.e., 11 weeks divided by M/n, where M is the total number of retrievals and n is the number of sources). Note that even when the average retrieval interval is the same, the actual retrieval points for each source are different under the two policies due to their different optimization approaches.

The vertical axis represents the retrieval delay of postings under each policy at the given resource constraint. More formally, it shows the *average delay*, which is the total delay $D(A)$ divided by the total number of postings generated by all sources, intended for easier interpretation by the readers.

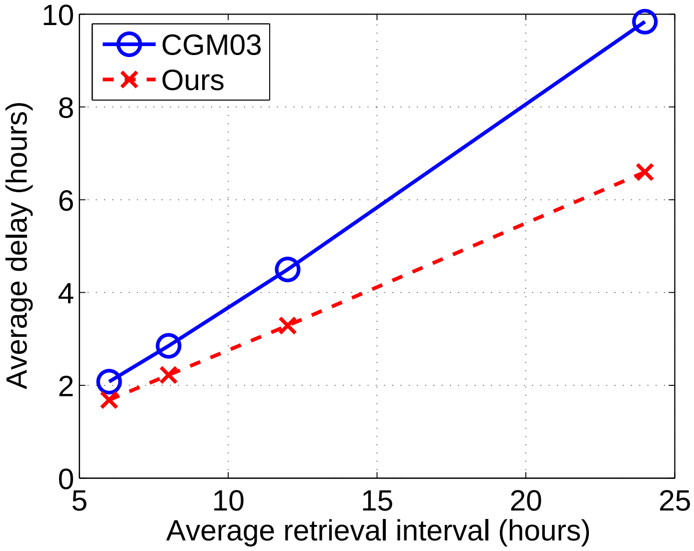


Figure 2.11: Comparison with CGM03 policy.

Figure 2.11 shows that the proposed policy clearly outperforms CGM03; in general, CGM03 gives a 35% longer delay than the proposed policy. Also note that the average delay is significantly shorter than half of the average retrieval interval, which is the expected delay when no optimization is performed. For example, when the average retrieval interval is 10 hours, the average delay is less than 3 hours under the proposed policy, which is more than 2 hours shorter than the 5-hour expected delay with no optimization.

Contribution of individual policy

To investigate further how much improvement we may get from each of the two optimizations (i.e., the resource-allocation policy in Section 2.3.1 and the retrieval-scheduling policy in Section 2.3.2), I now compare the average delay of the following four policies:

1. *Uniform scheduling*: Neither the resource-allocation nor the retrieval-scheduling policy is employed. That is, all sources are retrieved the same number of times, and the retrieval points are scheduled at uniform intervals. This can be considered as the baseline.

2. *Retrieval scheduling only*: Only the proposed retrieval-scheduling policy is employed. That is, all sources are retrieved the same number of times, but the retrieval points are optimized based on their posting patterns.

3. *Resource allocation only*: Only the proposed resource-allocation policy is employed. That is, depending on the posting rates of the sources, different numbers of times are allocated for each source, but the retrieval points are scheduled at uniform intervals for each source.

4. *Combined*: Both of the proposed policies are employed. The sources are retrieved different numbers of times and the retrieval points are further optimized based on their posting patterns.

Again, the first two-week data is used to learn the posting rates and the posting patterns, and the remaining 11 weeks are used to to simulate the retrieval policies and to compute the average delay. Every source is assigned an equal weight in this experiment.

Average retrieval interval	6hr	8hr	12hr	24hr
Uniform	180	256	352	645
Retrieval scheduling	159	211	310	518
Resource allocation	109	145	217	433
Combined	101	133	197	395

Table 2.2: Performance of 4 retrieval policies under different resource constraints.

Table 2.2 shows the average delays (reported in minutes) for the four policies under different resource constraints (from one retrieval per every 6 hours per source up to one retrieval per every 24 hours per source). For example, the number 180 in the second column of the second row means that the average delay is 180 minutes under the uniform policy when the average retrieval interval per source is 6 hours.

As shown in the table, the average delay under the uniform policy is close to half of the average retrieval interval. For example, when the average retrieval interval is 6 hours, the average delay under the uniform policy is 180 minutes (or 3 hours). This result is expected because when the postings are retrieved every 6 hours from a source, the maximum delay will be 6 hours and the minimum delay will be 0 hours, with the average being 3 hours.

The results also show that both resource-allocation and retrieval-scheduling policies are effective in reducing the average delay. For example, when new postings are retrieved once every 24 hours on average (the last column), retrieval scheduling and resource allocation decreases the delay by 20% and by 32%, respectively, from the uniform policy. Combined together, a 40% reduction is observed in the average delay compared to the uniform policy.

While both the resource-allocation and retrieval-scheduling policies are effective in reducing the average delay, the improvements are observed to be obtained through different mechanisms. Under the resource-allocation policy, resources are taken away from the sources of low posting rates (or of low importance) and are allocated to the sources of high posting rates (or of high importance). Thus, while the *average* delay is decreased, it ends up *increasing* the *maximum* delay of postings (for the sources of low posting rates). In contrast, the retrieval-scheduling policy improves the delay simply by selecting the best retrieval time for a source without reallocating resources among the sources, so the maximum delay do not vary much among the sources under this policy. For example, under the resource constraint of one retrieval per day per source, the maximum delays of a posting was 1440 minutes for the retrieval-scheduling only policy, while the maximum delay was 9120 minutes for the resource-allocation policy. Given this result, employing only the retrieval-scheduling policy is recommended when a tight bound on the maximum delay is important.

2.4.3 Evaluation of policy under miss penalty metric

In this section, I evaluate the performance of the proposed policy under *miss penalty* metric. Note that the *miss penalty* metric enables optimization based on both the data posting pattern and the user-access pattern. In order to measure the improvement from these two optimizations separately, the results from the following three policies are compared:

- *Uniform* - a simple method that schedules retrievals evenly spaced. For example, with 3 retrievals per day, all are spaced 8 hours apart. The majority of client-side RSS-feed readers employ this policy.

- *Data only* - the retrieval-scheduling method that optimizes based on the *miss penalty* metric by considering only the posting patterns of data sources and assuming a constant user-access pattern.
- *User+data* - the retrieval-scheduling method that optimizes based on the *miss penalty* metric by considering both the posting patterns of data sources and the user access pattern.

Dataset collection

The objective of this experiment is to show that, when user-access patterns are considered in retrieval scheduling, the number of articles missed by users can be further reduced when compared to using data posting patterns alone. Given the difficulty of obtaining actual user-access patterns from online content aggregators, I solicit individual users to supply their own user-access patterns using a browser plugin.

The user-access patterns were collected from nine volunteers from staff members of the NEC Labs America and students of the UCLA CS department who have installed a firefox browser plugin that records the time, the content, and the referring URL of the Webpages[8] a user has browsed. Their browsing activities were collected for 3 consecutive weeks to be used as the user-access patterns to compute the optimal schedules and to evaluate the *miss penalty* experienced by users.

Note that due to the small sample size and the inherent bias in the user selection, the findings in this experiment may not be conclusive enough to predict the

[8] Certain Webpages, such as those starting with *https* in the URL and those coming from well-known Web-based e-mail services, are not captured. Users are also given the option to exclude Webpages from certain domains from being captured due to privacy concerns.

performance gain in a more general setting. Thus, the primary goal of this small pilot study is to obtain a preliminary indication on the potential improvement brought by retrieval scheduling under the *miss penalty* metric when user-access patterns are considered. Interestingly, as I report later in Section 2.4.4, it is observed that the aggregated user-access pattern among the nine sampled users and the Web usage activities captured from a much larger pool of users both exhibit a similar pattern. Therefore, it is my expectation that a general online RSS aggregator is likely to benefit from the proposed policy as well, even though the degree of improvement may be significantly different from what I report here.

The user patterns were collected in 2007. In order to match the same time frame, the postings from a collection of 1.5K frequently updating RSS feeds published in the `blogger.com` domain were downloaded at the same period of time to be the data posting patterns when evaluating the proposed scheduling algorithm; hence, it is different from the one described in Section 2.4.1. Despite the difference in collection time, their composition and characteristics are similar, with a power-law like posting frequency distribution and fluctuating yet repeating patterns. Authors on `blogger.com` come from different parts in the world and the posting update times specified in the RSS feeds are expressed in their respective time zones, in both the experiment and graphs shown in this section, they are normalized to the Pacific Daylight-savings time (PDT).

Performance evaluation

The data posting patterns and the user-access pattern are computed based on the data collected in the first two weeks. The posting patterns are clustered into 20 groups, and the class centroids are used as representatives when computing their optimal schedules respectively. (Samples of common posting patterns are shown in Figure 2.12.) After the retrieval schedule is derived, the trace of data posting times and user-access times in the third week are used to compute the *miss penalty* under different resource constraints (from 2 to 5 retrievals per RSS feed per day).

For every user access, the *miss penalty* (the number of postings missed among the 1.5k RSS feeds) is counted to evaluate the performance. Figure 2.13 shows the comparison of the three methods. To make it easier to comprehend the numbers, the *miss penalty* metric is scaled with respect to the performance achieved by the *uniform* schedule and average the percentage reduction of *miss penalty* over the 9 users. It is observed that, in general, *user+data* reduces 40% of the number of postings missed as compared to *uniform*; when compared to *data only*, it reduces roughly 25% on average.

2.4.4 Learning posting rates, posting patterns, and user-access patterns

In order to implement the resource-allocation and retrieval-scheduling policies, the aggregator has to learn the average posting rate and the posting pattern $\lambda(t)$ of each source. Assuming that they do not change rapidly over time, they may be estimated by observing the sources for a period of time and the estimations are used to determine the optimal monitoring policies.

Measuring the posting rate can be done simply by counting the total number of postings generated within a particular learning period and dividing it by the length of the period. Learning the continuous posting pattern $\lambda(t)$ is more difficult, because only discrete events of posting generation are observed. Therefore, we first count the number of hourly postings at every source and build a daily histogram of hourly postings for the sources. We then overlap the daily histograms

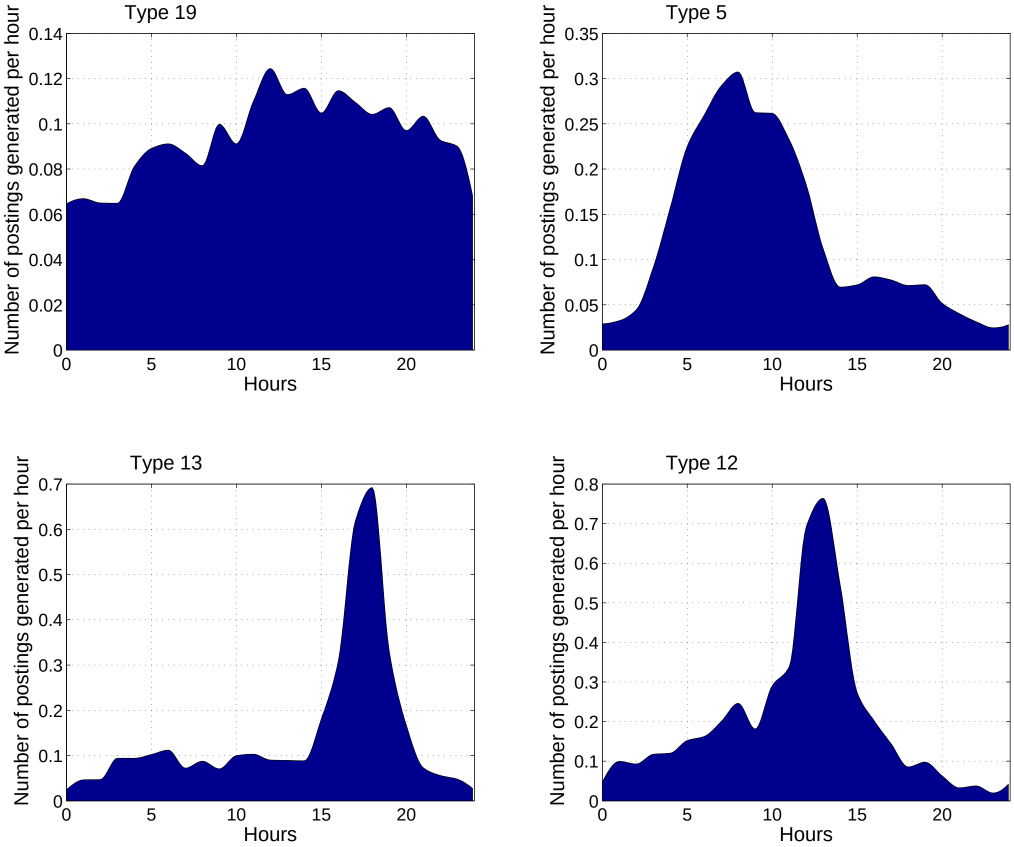


Figure 2.12: Samples of data posting patterns.

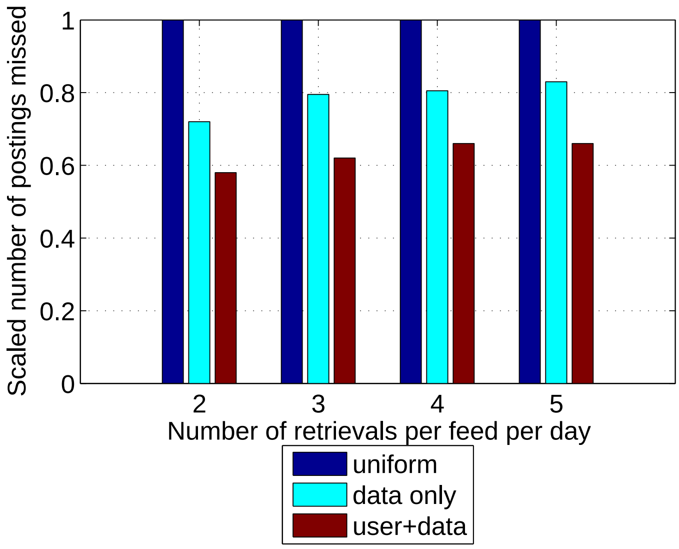


Figure 2.13: Comparison of three methods under the *miss penalty* metric.

for k-week data for each source and obtain a graph similar to Figure 2.16. This discrete posting histogram can then be smoothed by a interpolation method and be applied in the algorithm described in Section 2.3.3.

Note that there exists a clear tradeoff in deciding how long should we choose to monitor a source to estimate $\lambda(t)$; if the length is too short, the estimated $\lambda(t)$ may be inaccurate due to the randomness in the posting generation. However, if it is too long and if the posting pattern itself changes over time, the estimated $\lambda(t)$ will become obsolete by the time it is obtained (making the monitoring policy based on the estimated $\lambda(t)$ ineffective). The length of the estimation period is referred as the *estimation window*. Later in the experiment section, we evaluate the impact of the length of estimation on the effectiveness of the policies and empirically determine the optimal estimation period.

Learning posting rates

I first study the optimal window length for learning the average posting rate λ_i for source O_i. To investigate this, at the beginning of each day, the past k-day history data is used to estimate the posting rate of each source and decide the optimal number of retrievals per day for each source. This process is repeated over the entire 3-month data and the average delay at the end of the 3-month period is measured.

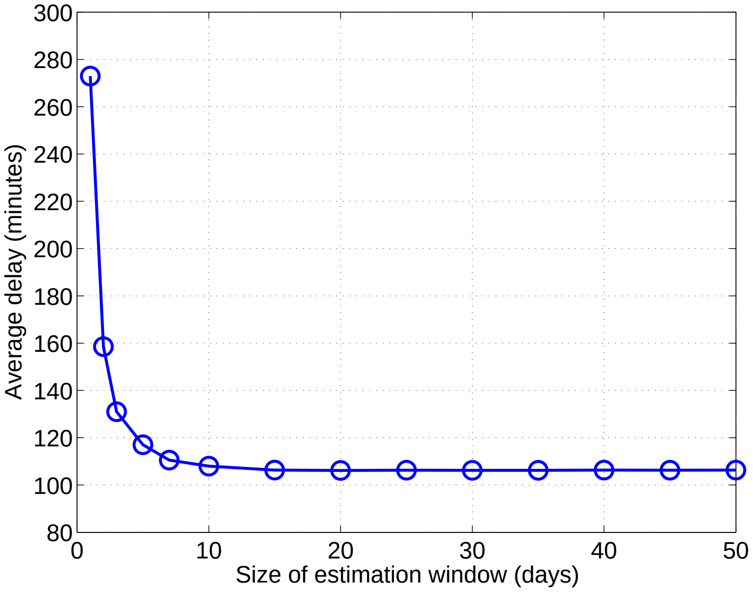


Figure 2.14: The effect of estimation window width.

Figure 2.14 shows the average delay of postings for different k values.[9] The graph shows that average delay decreases as the estimation window gets longer, which indicates more accurate estimation of the posting rates. However, there is no more improvement beyond k =14, which suggests that a 14-day worth of data is adequate to smooth out fluctuations and get reasonable estimates.

[9] The graph is obtained when postings are retrieved 4 times per day per feed on average. The results were similar when different numbers of retrievals per day are used.

In addition, the fact that the delay does not increase after $k = 14$ suggests that the posting rate of a source is stable and does not change significantly over time. To further investigate the change in the posting rate, we plot the following graph: the posting rate of each source using the first 14-day trace is calculated and used as the x-coordinate. The posting rate based on the last 14-day trace is again calculated and used as the y-coordinate. Based on the two coordinates, an x-y scatter plot is drawn and shown in Figure 2.15. In this graph, if the posting rates are stable, all dots should be aligned along the diagonal line $y = x$. Different colors for the dots are used depending on their proximity to the diagonal line.

- *Group A (darkest)*: the top 50% of dots that are the closest to the diagonal,
- *Group B (lightest)*: the top 50%–90% of dots closest to the diagonal, and
- *Group C (medium light)*: the rest

In the graph, it shows that most of the dots are very close to the line $y = x$; more than 90% of the dots are tightly clustered in a narrow band around $y = x$. This result indicates that the posting rates of most sources are stable, at least within the RSS sources that are monitored in the experiments.

Learning posting patterns

I now study the optimal window size for learning the posting pattern $\lambda(t)$. For this task, the number of hourly postings at every source is counted and used to build a daily histogram of hourly postings. The daily histograms for the k-week data for each source is then overlapped and a graph similar to Figure 2.16 is obtained, which is used as the $\lambda(t)$ graph of the source. Different k values are used to obtain this cumulative count graph. The retrieval-scheduling policy is then applied, and the average delay is measured for each k value setting. The

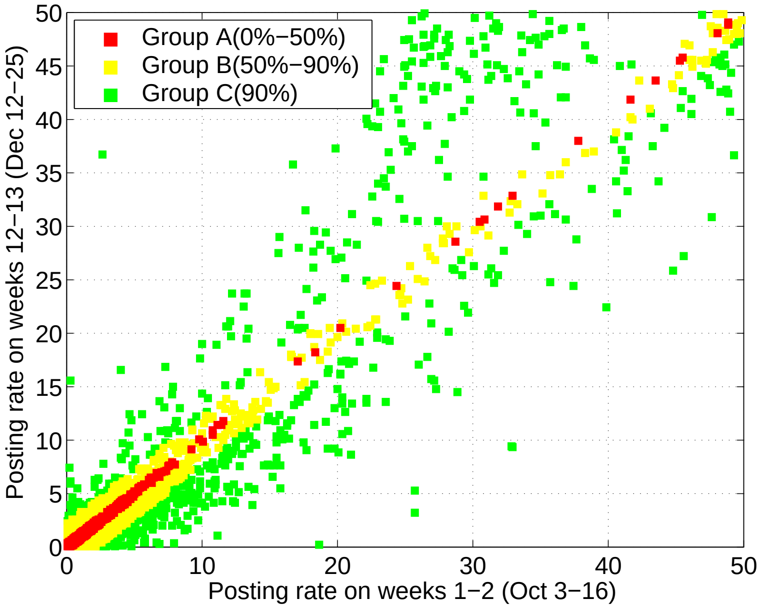


Figure 2.15: Correlation between posting rates measured at different times.

result of this experiment is shown in Figure 2.17, with the x-axis showing the k value used to obtain the $\lambda(t)$ graph and the vertical axis showing the average delay at the given k value. The graph shows that the size of k does not affect the final delay too much, which indicates that the accuracy of the posting pattern estimation is not affected by the estimation window size much. Given this result and the result from the posting rate estimation, a past 14-day history data seems to be a good choice in learning both the posting rate and the pattern of each source.

Learning user-access patterns

I use a similar technique that learns the posting pattern $\lambda(t)$ to estimate a user's access pattern $u(t)$ based on the data collected by the browser plugin. Fig 2.18 shows four samples of the user-access patterns obtained from the volunteers.

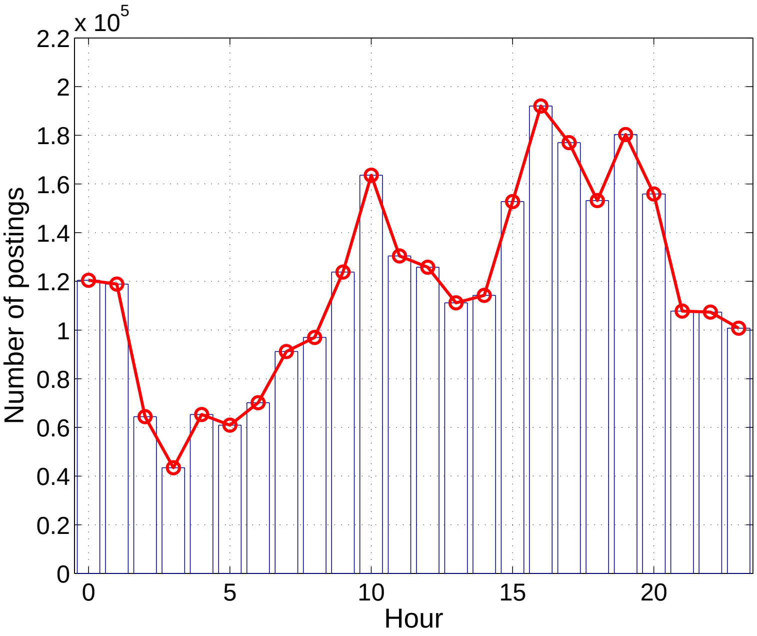


Figure 2.16: Aggregated posting pattern of 5,566 RSS feeds.

They all demonstrate the periodicity as described in Section 2.2.1. The daily access activity overlapped over 2 weeks closely resembles a user's work schedule. The proposed algorithm can exploit such fluctuations to devise tailor-made crawl schedules that prevent users from missing fresh content.

Besides the individual user-access patterns, I also present their aggregated pattern in Figure 2.19, which also shows a similar fluctuating pattern but at a lesser degree because they are combined from multiple users. In addition, Figure 2.20 shows the daily Web search activities to Google captured among all users in the UCLA CS department network over a 7 month period in 2004 with a total of 237k searches. The two patterns illustrate that the aggregated user-access pattern, either from a lesser number of users or from a much larger population, both experience similar behavior. I believe the retrieval scheduling policy under *miss penalty* metric can achieve similar level of performance improvement, as in Sec-

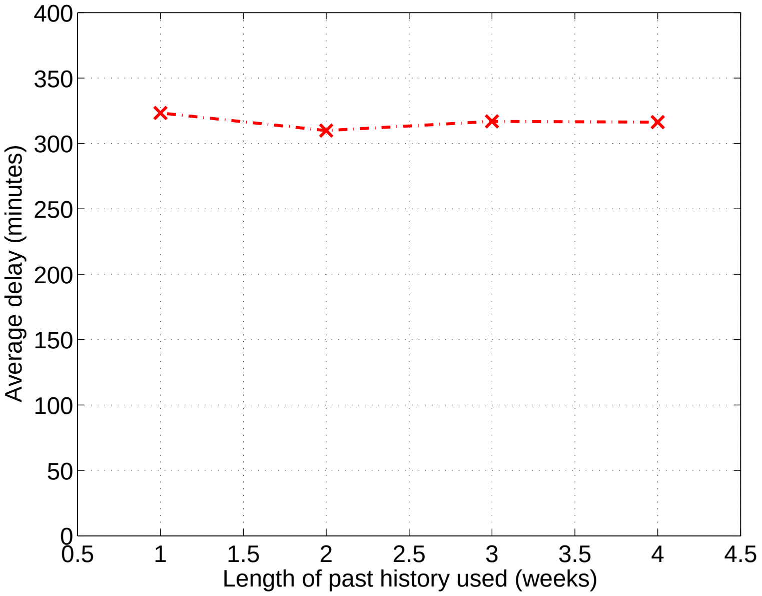


Figure 2.17: Effect of different learning periods of posting patterns.

tion 2.4.3, when online content aggregators use the aggregation of all subscribers to an RSS feed as the user-access pattern.

When retrieval schedules are optimized based on the user-access pattern, which is obtained from past history, it is important for the pattern to be predictable. To investigate this issue, I compare the correlation between the hourly number of Webpage accesses across two consecutive days in the 2 weeks' browsing activities obtained. Figure 2.21 shows the correlation of all the 9 users' hourly Webpage access rates 24 hours apart. Suppose a user accesses 10 Webpages between 3pm and 4pm on day 1, and she accesses 15 Webpages between 3pm and 4pm on day 2. Her activity will be recorded as a point with coordinate $(10, 15)$ in the graph. The points are then sorted according to their proximity to the diagonal line $x = y$, where the closer the points are to the diagonal, the more predictable the access pattern is. To better visualize the data, the strata of points that are

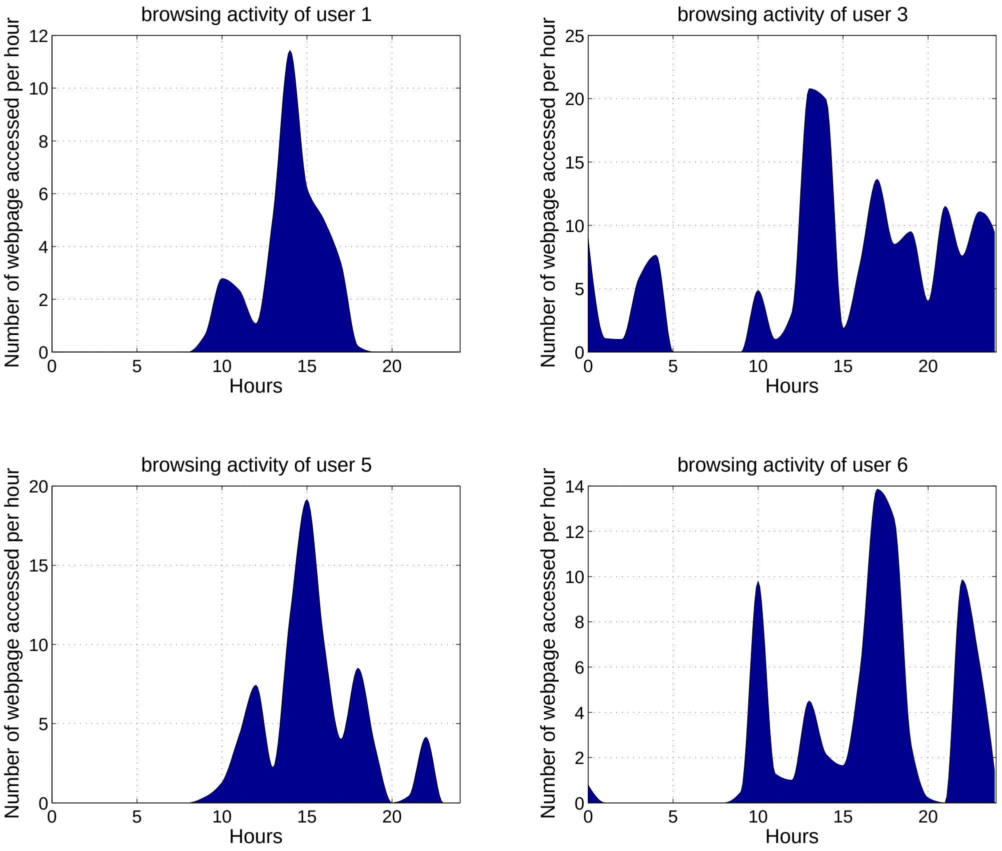


Figure 2.18: Samples of user-access patterns.

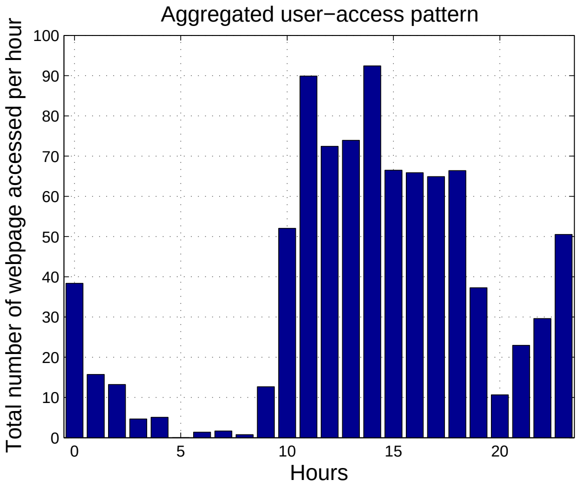


Figure 2.19: Aggregated user-access pattern of nine users.

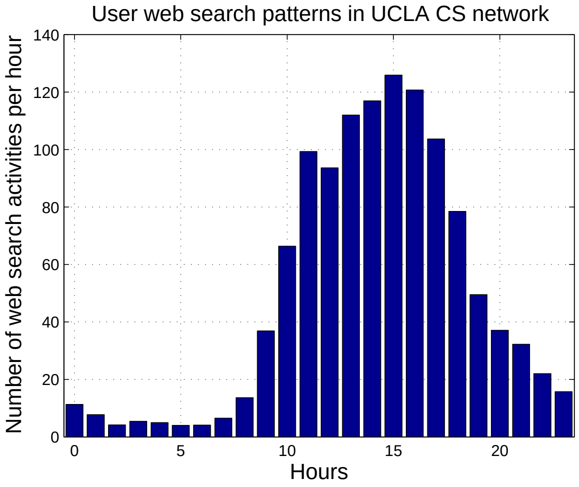


Figure 2.20: Aggregated user-access pattern from UCLA CS departments Google traffic.

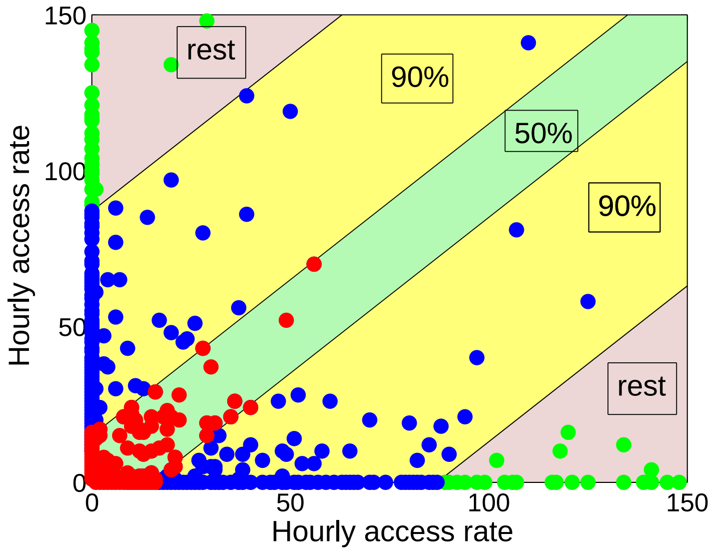


Figure 2.21: Correlation of user-access rate in consecutive days.

within 50% to 90% proximity to the diagonal are shown in different colors. From this observation, the user-access pattern is not as predictable as the posting rate shown in Figure 2.15; thus, a shorter history is recommended for modeling the user-access pattern in order to adapt to changes more quickly.

2.4.5 Potential saving by push-based approaches

Other than the pull-based approach that I have mainly investigated in this chapter, there can be a push-based approach where the data sources notify the aggregator whenever a new posting appears. Under this approach, the aggregator no longer needs to poll the sources regularly or maintain the posting-pattern profile of each source. Furthermore, since only new postings can be pushed to the aggregator, no resource will be wasted retrieving previously downloaded postings. In the dataset, it shows that, on average, each RSS feed contains the 12 most recent postings on average.[10] However, each feed only generates 4.3 new postings per day on average; therefore, if the aggregator retrieves from the data sources once a day under the pull-based approach, about 7.7/12 = 64% of the bandwidth will be wasted in retrieving previously downloaded postings, which will be saved when a push-based approach is employed. Furthermore, if implemented correctly, a push-based approach can potentially eliminate any noticeable delay of new postings at the aggregator, which is very difficult to achieve under the pull-based approach. For example, given the experimental results, if we want to achieve the average delay of less than an hour, the aggregator needs to contact the sources at the average rate of once every three hours under the pull-based approach, which corresponds to 8 retrievals per day. In comparison, a push-based approach will deliver a new posting to the aggregator only 4.3 times per day on average given their posting generation rate, which is roughly a 50% reduction in the number of retrievals per day.

These estimates show that a push-based approach is clearly beneficial to the aggregator in saving both bandwidth and the number of retrievals. However, it remains to be seen how widely a push-based approach will be deployed on the Internet, given the dominant adoption of the existing pull-based protocols and a number of potential problems that a push-based approach entails (such as the problem of spamming by certain RSS feeds that generate bogus spam postings in order to be shown more prominently at the aggregator and the problem of the additional cost at the sources for maintaining the list of subscribed aggregators and their preferences).

[10] The majority of implementation is to return either the 10 or 15 latest postings.

2.5 Related work

Web crawling is a well-studied research problem. References [CG00, CG03a, CG03b, CN02, CLW98, DKP01, EMT00, GE01, PRC03] investigate the problem of maintaining a fresh copy of Webpages for search engines. While the general problem is similar, the exact model and overall goals are significantly different from ours. For example, references [CG00, CG03a, CG03b, CLW98, PRC03] assume the homogeneous Poisson model to describe Webpage changes (which does not consider the fluctuations in the change rate as discussed in Section 2.2.1). References [BNW03, CG02] investigate the problem of minimizing the time to download *one snapshot* of the Web by efficiently distributing the downloading task to multiple distributed processes.

In more recent work [PO05, WSY02], researchers have proposed new crawling strategies to improve the user satisfaction for Web search engines by using more sophisticated goal metrics that incorporate the query load and the user click-through data. Since this body of work mainly focuses on getting improvement by exploiting the *user behavior* in the context of a Web search, it still assumes a relatively simple model to predict the changes of Webpages. For example, reference [PO05] assumes that Webpages always change in the same manner after every refresh. I believe that a more sophisticated change model such as the periodic inhomogeneous Poisson process can further improve the results of these studies and is complementary to this body of work.

In terms of improving the user's browsing experience in the time perspective, pre-fetching is a technique commonly used to reduce the wait-time of loading Webpages. Such a technique can be deployed at different locations on the Web. In particular, when deployed on the client side [EGS98], pre-fetching algorithms predicts the links on the current page that are likely to be accessed by the user in the future and requests them in advance; hence, it reduces the waiting time of the user when loading pages. In the case of deployment on the server-side[Mar96, YZ01], pre-fetching works by analyzing the Web access log for document access patterns; it then pre-fetches subsequent documents from disk into main memory to reduce the access time when they are requested by the clients afterwards. Similar techniques can also be deployed on proxy servers [DPS04] that serve a collection of users within a subnet.

In the context of a relational database, reference [GE01] has studied the use of periodic inhomogeneous Poisson process (referred to as Recurrent Piecewise-Constant Poisson process in the reference) to model record updates in a database. Due to the difference in the general goal and user requirements, however, its overall problem formulation and final solutions are significantly different from ours.

Researchers [DKP01, OW02] have also studied a source-cooperative approach where data sources actively *push* new changes to the aggregator. While these push-based approaches have significant benefits, whether they will be widely adopted for the general Web remains to be seen because it requires additional adminstrative work for publishers to adpot such scheme.

There have been recent efforts to make Web crawling more efficient by improving the underlying protocol. For example, Google sitemap protocol [good] allows a site administrator to publish the list of pages available at her site at a predefined location together with the last modification dates of the pages. While this new protocol helps a crawler discover new Webpages and their changes more efficiently, it is still based on the *pull* architecture, where a Web crawler is still responsible for periodically contacting the sites and downloading changes. Therefore, even if this protocol is widely deployed, the proposed monitoring policy will still be helpful in reducing the retrieval delay of new postings.

Researchers have studied publisher-subscriber systems [AF00, CDT00, FJL01, LPT99, SDR03, YG00] and proposed strategies for the efficient dissemination of information in these systems. This body of work mainly focuses on the efficient filtering of the overwhelming incoming data stream against a large pool of existing subscriber profiles and on the efficient data delivery method in the Internet scale; different from this body of work, our aggregator is not passively waiting for new data to come in; instead, the aggregator monitors and actively pulls from different data sources to collect new postings.

Researchers in the information-retrieval communities have also tackled the similar problem of monitoring multiple data sources but from the perspective of improving information relevance, which complements the information freshness issue addressed in this chapter. Under the federated search and P2P search framework, references [LC05, NF03, SC03] have proposed algorithms for querying a subset of data sources while maintaining a high quality of search results. Reference [Hos05] has provided a theoretical study on the trade-off between wait-time in querying the underlying data sources and providing a reasonable quality of results to users. In terms of efficient content delivery, reference [TIK05] has extended the publish/subscribe model on the DHT network that considers the data and language model in the data-placement process.

Google Alerts [Goob] and a number of blog aggregation services [Blob, Bloc] provide ways for users to subscribe to a set of news sources and get notified of any new articles. Unfortunately, the details of their implementation are a closely guarded secret. Besides, many interesting WWW applications, such as blogging and semantic Web [KQ04], semantic information retrieval based on XML data [CPC06], and the recently emerged Mashup [mas06] technology, can all benefit from a more efficient dissemination and exchange of content in the XML/RSS format. I believe that this work can be helpful to further improve these systems by providing the formal foundation and a disciplined solution to the delay-minimization problem in order to provide timely service.

2.6 Summary

In this chapter I have investigated the problems related to an RSS aggregator that retrieves information from multiple RSS sources automatically. In particular, I have developed a new RSS monitoring algorithm that exploits the non-uniformity of the generation of new postings and user-access patterns and is able to collect and deliver update content to users using minimal resources. The results have demonstrated that the aggregator can provide news alerts faster than the best existing approach under the same resource constraints. In addition, based on the analysis of the collected RSS data, it suggests that the posting rate follows a power-law distribution and that the posting rate and pattern remain fairly stable over time. A pilot user study also demonstrated the potential improvements when user-access patterns are taken into consideration. Finally, I have estimated the potential benefit of a push-based approach to the aggregator by measuring its savings in bandwidth and the number of retrievals per day.

The ability to provide timely information to Web users is of high commercial value to a Web service provider in both attracting user traffic and mining user behavior. I believe that existing RSS aggregators and blog search engines will benefit from the proposed monitoring policy to provide up-to-date information to users.

CHAPTER 3

Ranking of articles

3.1 Introduction

With the growing amount of user-generated content, it becomes difficult for a user to keep up with every piece of news closely, even if she only subscribe to a handful of RSS sources using an online content aggregator. In view of this, it becomes an issue for online content aggregators to rank new articles among different information sources subscribed by a user based on her reading interest. The top ranked list of articles is then presented to users as recommendations, in hope of improving user experience by filtering the largely irrelevant or uninteresting articles. Finding a good ranking function based on a user's interests is always a challenging issues in personalization recommendation systems. Such systems intend to provide a user with recommendations on news articles, documents, or products that are tailored toward the user's personal interests. They are used extensively in news portals (e.g., CNet and Google's personalized news search), RSS aggregators (e.g., Google reader), and e-commerce Websites (e.g., Amazon and Netflix's recommendation lists). In these systems, a key research issue is how to capture the users' interests effectively in order to rank the items.

Many important issues, such as the scalability of the system and the effectiveness of the recommendation, rely on how accurately we can capture a user's interests, which are not usually stated explicitly, such as by a subscription list that is going to be described in Chapter 4. In general, different forms of personal information manager rely on ranking pieces of information from a large knowledgebase, which can be news articles, technical documents, or even gossips in online social applications, to serve as an information filtering process. Although the settings may be different from a server-based aggregation model described in Chapter 1, they share the same motivation to provide better personalized recommendations through mining a user's interests based on her interaction with a large pool of information sources. Some challenging issues in capturing user interests are listed as follows.

Unobtrusive and quick detection

It is obtrusive to solicit users about their interests explicitly, though the result may be more accurate. To avoid being obtrusive, most real applications adopt implicit feedback by, for example, checking whether a user purchases a given product. However, such passive methods usually capture user interests very slowly, because whether a user makes a purchase cannot be controlled by the recommendation system. The challenge here is how to get user feedback quickly in an unobtrusive way.

Focused vs. diversified recommendation

Traditional recommendation systems, such as collaborative filtering systems, emphasize measures such as accuracy, precision, or recall. To score high in these measurements, systems usually recommend items with very focused topics in order to cater to users' main interests. However, users' satisfaction depends on both precision and diversification [ZMK05]. That is, a recommendation list that matches a user's multiple interests is preferable to a list with a single topic, although the single topic may be the one in which she is mostly interested.

Drift of user interests

A user's interests are likely to change over time. For example, soccer fans may temporarily lose interest in the World Cup when the World Cup is finished. As a matter of fact, the drift of user interests (concept drifts) has been designated as one of the top four machine learning challenges for user modeling [WPB01]. As a result, a system that captures user interests should adapt itself to the changes of user interests as well as to the emergence of new user interests.

To solve the above challenging issues, I, together with my collaborators, propose a learning framework and an algorithm to rank articles by actively capturing user interests through an unobtrusive interactive recommendation process. Unlike a naive greedy algorithm, which only *exploits* the model of users' interests, the proposed algorithm takes into account *exploration*, i.e., discovering user potential interests, as well. Therefore, the proposed algorithm can discover diversified recommendation while it focuses on users' main interests. Due to the exploration, the algorithm can quickly adapt to the drifting of user interests as well.

The rest of this chapter is organized as the following. In Section 3.2 we build a model for ranking articles based on user behaviors. In Section 3.3 we describe the details of our algorithm. A pilot user study to demonstrate the idea and effectiveness of our proposed method is given in Section 3.4. Finally, in Section 3.6, we provide a summary and future directions after the discussion of related work in Section 3.5.

3.2 User model

As depicted in Figure 1.3, an online content aggregator proveds a user with a ranked list of articles that are tailored toward her personal interests. The system observes the user's activities while she reads through different articles and recommends the user a list of new articles, which are often blog postings. Assuming that *clicking* the link to a recommended item after *reading* a short description of it indicates that she likes the *topic* of the item recommended, the system learns a user model from this observation and better ranks the recommendation list in the future.

In this section, we model the user-interests component in the recommendation system. We assume that user's interests are represented by a combination of K general topics. Let the K topics be $\{\text{topic } 1, \cdots, \text{topic } K\}$, where K could be a large number. We further assume that each item, that can potentially be recommended, belongs to only one topic to simplify the model and the analysis.

When the description of an recommended item of topic i is read by the user, the user clicks the link to the recommended page with probability θ_i, that is,

$$\theta_i = \Pr(\text{click}|\text{read}, \text{topic } i). \tag{3.1}$$

Then the user's interests can be represented by the parameter $\Theta = \{\theta_1, \cdots, \theta_K\}$, which is going to be estimated.

When a recommendation item is shown to the user, it has different chances to attract user attention, depending on its position in the ranked list. In the Web search-engine community, it is well-known that the position of an entry in the query result list heavily affects its chance to be clicked by the user[ABD06b]. We capture this phenomenon by a probability model; we denote the probability that the user read a recommendation at the j-th position of the list as

$$g(j) = \Pr(\text{read}|j), \text{for } 1 \leq j \leq K. \tag{3.2}$$

Of course, we do not expect the content of all K topics to be in the list, which means that $g(j) = 0$ for j greater than the length of the list. In addition, we assume the probability, $g(j)$, is independent of the user's interest and usually decreases monotonically as j increases.

Let variable $R = \{r_1, \cdots, r_K\}$ be the ranking order given to the K topics. Then we can write the utility of this ranking order as

$$U(R; \Theta) = \sum_{i=1}^{K} \Pr(\text{click}|\text{read}, \text{topic } i; \theta_i) \Pr(\text{read}|r_i) = \sum_{i=1}^{K} \theta_i \cdot g(r_i). \tag{3.3}$$

Given a user model, Θ, the goal of the system is to maximize the utility, $U(R, \Theta)$.

3.3 Learning user profile by exploitation and exploration

To maximize the one-step expected utility, we recommend and rank topics according to their expected utilities, $\{\theta_i \cdot g(r_i)\}$ — the probability that the user reads the recommendations times the probability that the user clicks the topics. This approach is to *exploit* the best estimation of the user model, Θ, to gain the optimal one-step utility. We call it the *greedy* approach.

Because the greedy approach uses the estimated user model, Θ, not the true user model, the overall utility usually is not optimal. Such an approach puts the best estimated θ's on the list, which may deprive the opportunity of showing the true optimal θ's. Without these being shown, we are unable to get an accurate estimate of the actual best θ's, which lowers the utility gain in the later steps. Showing the topics of non-optimal estimated θ's is known as *exploration.* Therefore, we face a trade-off between exploitation and exploration to gain the optimal overall utility. This was also illustrated in the well-known multi-armed bandit problem [GJ74].

We assume the prior of θ_i follows the beta distribution, $B(\theta_i|\alpha_i, \beta_i)$, where α_i and β_i can be initially set to some fixed constants. The reason for using the beta distribution is that the beta distribution is a conjugate distribution of the Bernoulli distribution (see [GCS95]), which gives us a posterior distribution in the same form. When the recommendation is ranked at r_i and is not clicked, we have

$$\begin{aligned} d\Pr(\theta_i|\neg\text{click}, r_i) &\propto [1 - \Pr(\text{click}|\theta_i, r_i)]\, d\Pr(\theta_i) \\ &= [1 - \Pr(\text{click}|\text{read}, \theta_i)\Pr(\text{read}|r_i)]\, d\Pr(\theta_i) \\ &= [1 - \theta_i g(r_i)]\, d\Pr(\theta_i) = [1 - \theta_i g(r_i)]\, B(\theta_i|\alpha_i, \beta_i) d\theta_i \\ &\approx [1 - \theta_i]^{g(r_i)}\, B(\theta_i|\alpha_i, \beta_i) d\theta_i \propto B(\theta_i|\alpha_i, \beta_i + g(r_i)) d\theta_i, \end{aligned}$$

in which we used the approximation $(1-x)^y = 1 - xy + O(y * x^2) \approx 1 - xy$. Since $B(\theta_i|\alpha_i, \beta_i + g(r_i))$ is normalized, we have

$$d\Pr(\theta_i|\neg\text{click}, r_i) \approx B(\theta_i|\alpha_i, \beta_i + g(r_i)) d\theta_i.$$

That is, the posterior distribution of θ_i follows $B(\theta_i|\alpha_i, \beta_i + g(r_i))$ if the user does not click the recommendation of topic i at position r_i in the list. If the recommendation of topic i is clicked, the posterior distribution of θ_i follows $B(\theta_i|\alpha_i + 1, \beta_i)$. Thus, we have the update formula for the distribution of each θ.

Given the distribution of each θ in parameter Θ, we design an algorithm that actively probes user interests by presenting a list of recommendations to the user. The algorithm should achieve two goals at the same time. The first goal is exploitation, that is, to recommend contents that the user is likely to click. The second goal is exploration, that is, to reduce the variance of Θ by presenting to the user those topics whose θ's have the largest variances and uncertainty. To achieve both goals, when ranking a given topic, we use its expected utility plus a term related to its variance instead of solely using the expected utility as in the greedy approach. The term related to the variance is known as the *exploration bonus* [Sut90]. For our case, given $\theta_i \sim B(\cdot|\alpha_i, \beta_i)$, the expected utility is $\alpha_i/(\alpha_i + \beta_i)$, and its variance is $\alpha_i\beta_i/[(\alpha_i+\beta_i)^2(\alpha_i+\beta_i+1)]$. We define the exploration bonus as the variance scaled by a weight parameter λ. Hence, the ranking score, a combination of the expected utility and the exploration bonus, is

$$\frac{\alpha_i}{\alpha_i+\beta_i}\left[1+\lambda\frac{\beta_i}{(\alpha_i+\beta_i)(\alpha_i+\beta_i+1)}\right].$$

Another benefit of our algorithm, which handles both exploitation and exploration in a comprehensive way, is that it adapts itself more quickly to user-interest drifts than the greedy algorithm. There are two factors to be considered in interest drifting — discovering new interests and forgetting old ones. In terms of discovering new interests, the exploration nature of our algorithm helps quickly pick up new user interests. The following example illustrates how our algorithm forgets old interests. Consider there are two topics, a and b, to be ranked and displayed in a one-item-at-a-time recommendation list, where $\alpha_a = 10$, $\beta_a = 1$, $\alpha_b = 1$, and $\beta_b = 1$. For the exploitation part, $\alpha_a/(\alpha_a + \beta_a) = 10/11$ and $\alpha_b/(\alpha_b + \beta_b) = 1/2$. If the user stops being interested in topic a, for the greedy algorithm, it takes 10 more rejections of a for topic b to replace a's position. However, since topic b has only been rejected once, it is not certain that topic a is superior to topic b. In our algorithm, when $\lambda = 3$, it takes only 4 more rejections of a for topic b to replace a's position.

3.4 Simulation and User Study

We use both simulation and a user study experiment to evaluate the performance of our recommendation strategy. The reason for using simulation is that user study may involve many hidden factors that are not fully captured by our model. Therefore, we use simulation as a better-controlled environment to demonstrate some properties of our proposed recommendation strategy (referred to as *Exploitation and Exploration*, *E&E* in short, hereafter). For comparison, we also implement two naive baselines:

- *random* - it presents topics in a random order, where every topic is shown to a user the same number of times on average.
- *greedy* - it ranks and presents topics based on their learned θ_i' respectively (i.e. $\alpha_i/(\alpha_i + \beta_i)$). Where displayed items always come from topics with the top j θ value, j is the size of the recommendation list.

The *random* method is a simple strategy without exploitation. However, as shown below, it shows good performance in terms of the accuracy of parameter estimation. The *greedy* strategy is a strategy without exploration, i.e., a special case of the *E&E* strategy with the exploration bonus weight λ set to 0. That is, the greedy strategy *exploits* user click history without actively *exploring* previously unknown user interests. By comparing the *E&E* strategy with these two baselines, we show that exploration does help in terms of quickly capturing diversified user interests, which may drift over time.

3.4.1 Simulation

Parameters of the simulation are shown in Table 3.1. The settings are to simulate

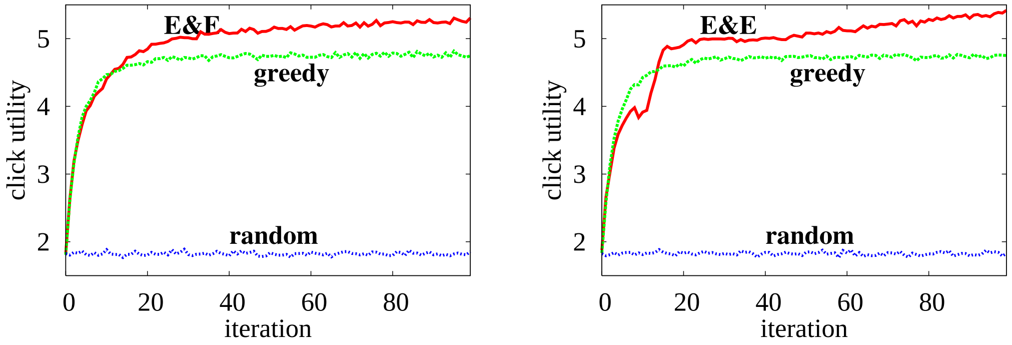
Total number of topics (K)	45
Size of display list (j)	7
Read probability ($g(j)$)	[1 0.95 0.9 0.85 0.8 0.75 0.7]
Initial α, β values	$\alpha = 1$, $\beta = 3$
Click probabilities of interested topics (θ's)	7 are uniformly distributed between [0.9,1] 13 are uniformly distributed between [0.4,0.7]

Table 3.1: Parameters used in simulation.

the environment of the user study described in Section 3.4.2. We simulate 2000 users, each for 100 iterations. $\Theta = \{\theta_1, \ldots, \theta_{45}\}$ consists of 7 large values and 13 medium ones, while the remaining 25 are zeros. We use this simulation to show that *E&E*, through exploration, can discover topics with the highest θ_i values while the *greedy* strategy may, instead, remain at suboptimal. We compare the performance of the three algorithms in three different aspects, namely, click utility, estimation accuracy, and tracking interest drift.

Click utility

Click utility is a commonly used metric for measuring user satisfaction on recommendation systems or search results [Hij99, QC06]. It can be considered as the *attractiveness* of display items to convince users to click. Figure 3.1 shows the average number of clicks per iteration for the three algorithms in the first 100 iterations of the simulation. For *E&E*, we used two different exploration bonus weights ($\lambda = 10$ and $\lambda = 20$, respectively). Both *E&E* and *greedy* outperform *random* because they learn and display items matching a user's interests to maximize the likelihood of being clicked. We observe that although *E&E* achieves a lower click utility initially, it scores higher values in a long run (after the 10^{th}



(a) $\lambda = 10$

(b) $\lambda = 20$

Figure 3.1: Click utility of simulated users. For *E&E*, different exploration bonus weights (λ).

iteration). This result indicates that in *E&E*, some exploration cost is paid in the initial phase to identify topics with potentially high θ_i value and that such exploration pays off in the long run.

Estimation error of Θ

Other than the click utility, we use the estimation error of Θ (user interest) to evaluate the performance of the three recommendation algorithms. At each iteration, we compute the normalized mean absolute error (MAE), which is commonly used in recommendation systems literature [HKT04], of the learned θ'_i as follows, $\sum_{i=1}^{K} |\theta'_i - \theta_i|/\theta_i$, where topics with $\theta_i = 0$ are omitted in the calculation. Figure 3.2 shows the change of estimation accuracy along the iterations. Generally, the estimation accuracy improves (i.e., the estimation error reduces) as the learning process proceeds, where *E&E* shows a performance in between the worst (*greedy*) and the best (*random*), depending on the λ value. This observation suggests that more exploration gives better estimation accuracy; however,

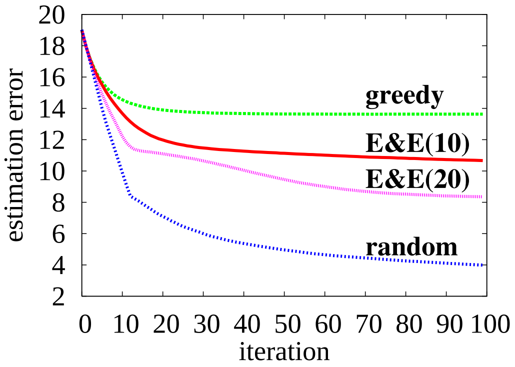


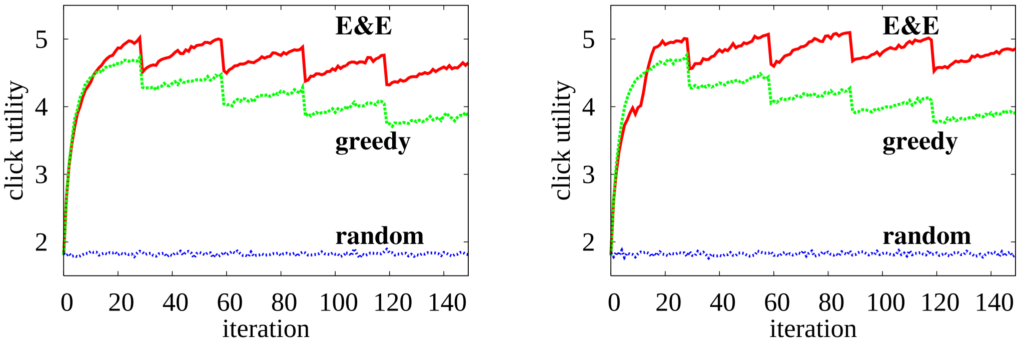
Figure 3.2: Estimation error of θ_i. For *E&E*, different exploration bonus weights (λ). *The lower estimation error is the better.*

although *E&E* gives better estimation accuracy, it does not sacrifice click utility, as *random* does.

Interest drift

The idea of exploration can be best illustrated by an interest-drift scenario: at every 30 iterations, we randomly replace 2 of the currently interesting topics with 2 that were previously uninteresting to the user. The simulation lasts 150 iterations with interest drift at the 30^{th}, 60^{th}, 90^{th}, and 120^{th} iterations. When a user drifts her interests to some less explored topics (higher variance of θ), *E&E* can capture these changes faster and improve the click utility accordingly. Figure 3.3 shows that, immediately after the user's interests drift, the click utilities of both *E&E* and *greedy* decrease due to the dominance of the previously learned profile. *E&E*, however, adapts to the drift much faster than *greedy* because of its exploration component, and the click utility rises back to the optimum level. Again, this effect is more significant when λ is larger.

From these controlled simulation studies, we can see that our *E&E* algorithm



(a) $\lambda = 10$

(b) $\lambda = 20$

Figure 3.3: Click utility of simulated users with different exploration bonus weights under interest-drift scenario.

outperforms the *greedy* algorithm, which exploits but does not explore. By combining both exploitation and exploration, our *E&E* recommendation strategy generates higher click utility, has lower estimation error, and tracks user–interest drifts more accurately.

3.4.2 User Study

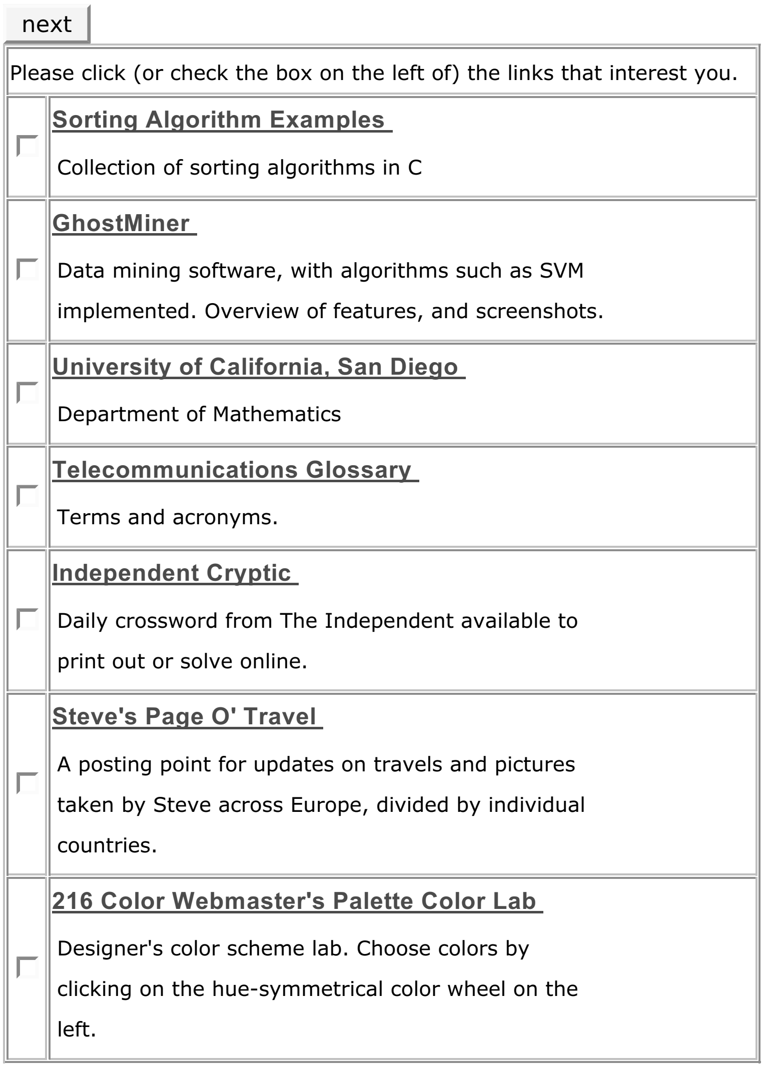
The simulation has revealed some properties of the *E&E* recommendation strategy; we now investigate the potential of the *E&E* strategy through a small-scale pilot study. As I will explain shortly, the findings from this experiment may not be generally valid due to the small number of participants in our experiments and the high concentration of their interest in technology field. Despite this shortcoming, it is our hope that this experiment will shed a light on the potential effectiveness of our strategy in a more realistic setting than simulation. We have recruited 10 users from the staffs of NEC Labs and students of the UCLA CS department to participate in the pilot study. In the experiment,

Arts/Animation	Arts/Architecture
Arts/Photography	Computers/Algorithms
Computers/Data_Communications	Computers/E-Books
Computers/Hacking	Computers/Human-Computer_Interaction
Games/Board_Games	Games/Game_Studies
Health/Beauty	Health/Mental_Health
Health/Occupational_Health_and_Safety	News/Analysis_and_Opinion
News/Magazines_and_E-zines	Recreation/Autos
Recreation/Travel	Science/Biology
Science/Math	Science/Physics
Shopping/Clothing	Shopping/Consumer_Electronics
Society/History	Society/Law
Society/Politics	Sports/Baseball
Sports/Cycling	Sports/Football
Sports/Squash	Sports/Tennis

Table 3.2: The selected topics from ODP.

we randomly select 45 categories and their respective Webpages from the Open Directory Project [DMO], which is a repository of pre-categorized Webpages, in level 2 of its hierarchy. A sample of which is shown in Table. 3.2 and a screen-shot of the interface is shown in Figure 3.4.

Before the experiment begins, we ask the users to indicate, on a scale of 1 to 9, their interest level on each topic as the ground truth (θ_i). This ground truth is not used by any algorithm but used only for computing performance. In each iteration of the experiment, the URLs of 7 Webpages, together with their titles and descriptions, are presented to the user. Each of them comes from a different category. The user is instructed to click on the URLs that interest her. When no



Please click (or check the box on the left of) the links that interest you.

☐	**Sorting Algorithm Examples** Collection of sorting algorithms in C
☐	**GhostMiner** Data mining software, with algorithms such as SVM implemented. Overview of features, and screenshots.
☐	**University of California, San Diego** Department of Mathematics
☐	**Telecommunications Glossary** Terms and acronyms.
☐	**Independent Cryptic** Daily crossword from The Independent available to print out or solve online.
☐	**Steve's Page O' Travel** A posting point for updates on travels and pictures taken by Steve across Europe, divided by individual countries.
☐	**216 Color Webmaster's Palette Color Lab** Designer's color scheme lab. Choose colors by clicking on the hue-symmetrical color wheel on the left.

Figure 3.4: User interface of the user study experiments.

more URLs are found interesting, the user will click the *next* button to proceed and get the next recommendation list. Each click by the user expresses a user's interest. At the same time, the fact that the user did not click some links in a list indicates that the user is not interested in those topics. Both pieces of information are used to update the α_i, β_i values of the corresponding topics.

We interleave 3 different strategies randomly throughout the iterations without informing the users to reduce the bias that can be introduced by different groups of users and ordering effects, while each strategy updates its parameters independently of the others. The experiment lasts 75 iterations, with 3 strategies run 25 iterations each. Such experiment settings minimize any potential bias by comparing the strategies without dividing users into groups or dividing the test into phases.

Click utility

Figure 3.5 shows the click utility (as the fraction of improvement of cumulative clicks over *random*) averaged over 10 users, with 3 users with an extremely small number of clicks dropped[1]. The behavior of each strategy is similar to the simulation prediction, where *E&E* performs noticeably better than *greedy*.

Estimation error of Θ

Since users indicate their interest level (ground truth) on a scale of 1 to 9, we map these levels to click probabilities (θ_i) by using $\frac{x-lb}{ub-lb}$, where x is the level selected, lb is the lowest level selected, and ub is the highest level selected by the user. We

[1] The reasons for low click utility were later found to be that 1) although the URLs do come from a user's interested categories, their quality is not high enough to be clicked, and 2) some topics are too broad to capture a specific interest (e.g., Sports-basketball encompasses various basketball pages instead of the well-known NBA).

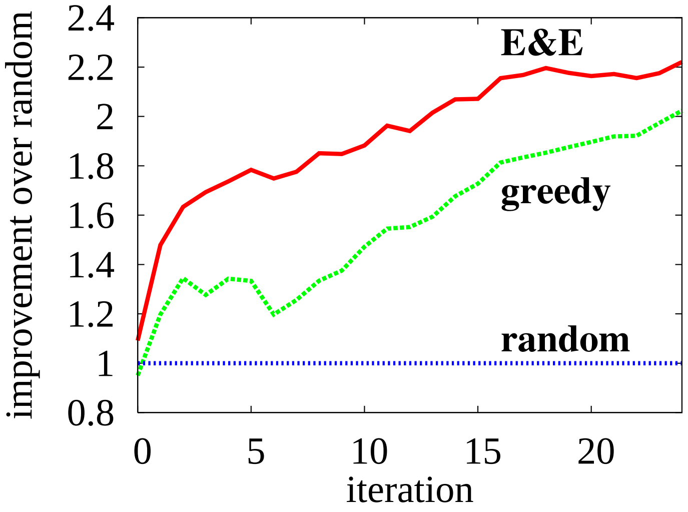


Figure 3.5: Comparison of click utility of *E&E*, *greedy*, and *random* strategies.

apply the equation for measuring estimation error and compute the values at the 25^{th} iteration of (i.e., the end of) the experiment. The estimation error values of *random*, *E&E*, and *greedy* are 21.6, 23.8, and 24.3, respectively. The relative performance is similar to the prediction from simulation; however, the absolute difference in performance is less noticeable. Even the *random* strategy cannot show a significant improvement in the estimation accuracy, which suggests that the interest levels indicated in the survey may not truly reflect the user's interest as captured by clicks. This discrepancy may be resulted from the mismatch of interest between users, which are mostly of Computer Science background, and the diverse range of topics selected. Nonetheless, we believe that the result averaging from multiple users gives certain support to our conclusion drawn from the simulation experiment that the accuracy of *E&E* in capturing user interest is in between of the *random* and the *greedy* strategies.

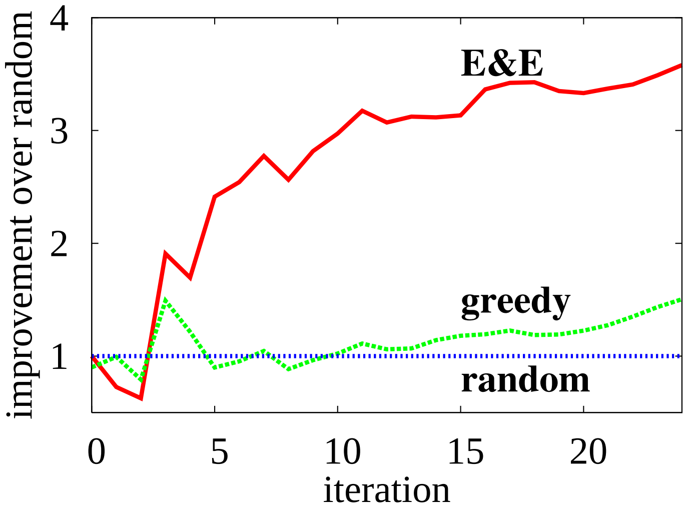


Figure 3.6: Comparison of click utility of *E&E*, *greedy*, and *random* strategies under interest-drift scenario.

Interest drift

In user study, it is hard to force a user to change her interests intentionally and record what has been changed. Therefore, to analyze interest drifts in user study, we design an experiment in which, after a user finishes the 75 iterations of her test, a different user (with substantially different interests according to her answers in the beginning survey) continues the test for another 75 iterations. This switch of users is used to simulate the case in which the user has interest drift at the end of the 25^{th} iteration. Figure 3.6 shows the average click utility (as the fraction of improvement of cumulative clicks over *random*) of 5 users, from the 26^{th} to 50^{th} iterations. The results show that *E&E* adapted faster than *greedy* and further improved the click utility towards the end in our pilot study.

While we cannot extend the results of this pilot study to general settings due to the inherent limitation of our experimental setup, the results confirm the conclusions drawn from the simulation study. That is, our *E&E* algorithm outperforms the exploitation-only *greedy* algorithm in terms of click utility, parameter estimation error, and adaptation to user-interest drift.

3.5 Related Work

In an invited talk at the 2005 User Modeling Conference, Jameson [Jam05] enumerated five major usability goals for user-adaptive systems as privacy, controllability, unobtrusiveness, breadth of experience, and predictability and comprehensibility. The personal information manager is designed to address the last three usability goals by the *E&E* recommendation strategy.

Learning users' interests and profiles has been studied extensively in various areas such as information retrieval, Web search and mining, etc. In the information-retrieval area, relevance feedback has long been used for improving the quality of retrieval [Eft00, KDF05]. A relevance feedback system is an interactive system in which queries are expanded by terms discovered from the documents that the user found useful, and therefore, it is a process of user-interest modeling. In the Web search and mining area, personalization has been one of the most important research topics [MCS00, MDL02]. In [Bro02, LLC05, RL04], the intentions of a user of a Web search engine are categorized into either navigational, where the user has very clear Website targets in mind, or informational, where the user's intention is to explore information on a topic, and various techniques are proposed to detect these intentions. In [ABD06a, ABD06b, QC06], techniques are proposed to incorporate personal interests into ranking algorithms to reflect a user's model where the user's interests are learned from the user's click history. Many other forms of information from users, such as display time [KB04] and user profiles [SHY04], have also been used to induce user interests, and various mathematical models, such as those in [JZM05] and [Joa02], have been proposed to describe user preferences. While these different approaches have proved effective in various areas, a key point that limits their flexibility is that they are all *passive* in nature. That is, all these approaches *exploit* historic data while ignoring *exploring* additional information about users. In comparison, user interests are *actively explored* in our approach.

In [SZ05], an active feedback framework is proposed for probing user preference where the documents to be probed are selected based on a statistical decision theory. However, there are two limitations in this work. First, it requires *explicitly* asking users for feedback, and second, it assumes that the ground truth, i.e., if a document is relevant, is available. In our algorithm, the active probing is achieved *implicitly.* Furthermore, we do not assume unambiguous ground truth and instead adopt a probabilistic model to describe the uncertainty in user preference.

3.6 Summary

In this chapter, we studied how to rank articles by capturing user interests effectively in a personalized recommendation system. We started by building a probabilistic model to model a user's interaction with the personalized recommendation system. Based on this user model, we proposed an algorithm in a learning framework that rank articles by actively capturing user interests through an interactive recommendation process. The proposed algorithm takes into account both exploitation, i.e., recommending items according to user interests that are currently known, and exploration, i.e., actively exploring potential user interests that are currently unknown. We designed simulation experiments to compare the performance of our algorithm with that of a random algorithm, which does not exploit, and a greedy algorithm, which does not explore. The experimental results demonstrated that by incorporating both exploration and exploitation, our recommendation algorithm achieved higher click utility, lower parameter estimation error, and more agile adaptation to user-interest drift. We have also carried out a small-scale pilot user study. While the results cannot be generalized due to the limitation of the experiment setup, the conclusion drawn is similar to those found in the simulations.

There are several future research directions. In this chapter, we assumed that each recommended item can match only a single interest of the user. This assumption can be too simplistic in practice, and currently we suggest to incorporate multiple topics to a single user click in our model as a future extension. Another future research direction is to study and incorporate dependency among items in the same list.

CHAPTER 4

Efficient personal recommendations

4.1 Introduction

The amount of user-generated content on the Web is exploding as the broadband Internet connection becomes ubiquitous and many "Web 2.0" services, such as Blogger, MySpace, Flickr, and *delicious*, make it extremely easy even for novice users to post and share their own creations on the Web. As discussed in Chapter 1, even an user subscribes to only a few of these information sources, it becomes very difficult for her to keep up with all the new information constantly being generated. To deliver better personalization, it becomes useful for online content aggregators to provide recommendations of popular topics (either as key phrases or links to Web resources) on the Web that users may not aware of to draw their interests.

These user-generated contents can be considered the expressions of individuals' opinions on various issues and items around them. For example, a technical blogger may write his opinion on a new and highly-anticipated gadget on his blog, and a *delicious* user may "bookmark" a page if she finds it worth a later visit. The participation of a large number of users in sharing their opinions on the Web has inspired researchers to build an effective "information filter" by aggregating these independent opinions. For example, *delicious* and Digg – two popular online bookmarking sites – count how many times a page is "bookmarked" or "digged" by their users and prominently show the most popular pages on their front pages. The hope behind these aggregation services is that by leveraging the independent judgment of millions of users, we can tap into the wisdom of the crowds and make a good collective judgment on what items and/or topics are truly interesting.

Such aggregation service is shown to be effective in finding the popular and interesting items from the Web. However, this approach suffers from two shortcomings: (1) a particular user's interest may be significantly different from the interest of the general public (2) it may be vulnerable to spam. For example, a group of users can collaborate to promote pages to the front page of Digg for their own benefit. Also, given the diverse groups of users on the Web nowadays, a Webpage that has been recommended by a particular group of users may not be of much interest to different groups of users.

In this chapter, to address the abovementioned problems, I explore the possibility of computing the *personalized aggregation* of the opinions expressed on the Web. Under our approach, I assume that each user indicates how much the user "trusts" each information source[1] — either explicitly or implicitly. Then as each source "mentions" or "endorses" different items[2] over time, the user gets a personalized recommendation of items based on how many times each item is endorsed by her trusted sources. By employing this "personalized" aggregation, it makes the recommendation more likely to align with the user's interest and less vulnerable to spam created by distrusted sources. In fact, in the experiments, it is observed that the proposed personalized aggregation approach indeed makes a

[1] An information source can be any independent source of information, such as a blog page maintained by a particular blogger or the bookmark page maintained by a particular delicious user.

[2] An item can be any potential object of interest, such as a Webpage mentioned on blogs or a hot topic or gadget being discussed on blogs.

significant difference in the items that are recommended and captures the user's interest much better than an overall aggregation service.

Although such an idea can be extremely effective, one important challenge in providing personalized aggregation is the problem of scalability. Given that there are potentially millions of users who may use the system and millions of sources to get the opinions from, computing the personalized aggregation every time a user requests causes significant computational cost, as we will see later. The main focus of this chapter is to address this scalability issue. Roughly, my idea for addressing this challenge is to model the personalized aggregation problem as a matrix multiplication and apply an effective matrix decomposition method to reduce the multiplication cost. As we will see later, the proposed approach for personalized aggregation reduces the computational cost significantly, often more than 75%, while the result of personalized aggregation is kept accurate enough.

The rest of the chapter is organized as follows. In Section 4.2, I formulate the *personalized aggregation* as the problem of weighted sum computation and present two basic methods to perform this computation. I then present a matrix representation of this problem and propose an efficient method to support *personalized aggregation.* In Section 4.3, I will describe the experiments with a real dataset to show the impact of personalization and the efficiency of our proposed method. Finally, I provide concluding remarks and a brief discussion of some possible future investigations in Section 4.5 after briefly going over related work in Section 4.4.

4.2 Framework

In my model, I assume that we want to compute personalized recommendations for n users ($U = \{u_1, u_2, \ldots, u_n\}$) by aggregating opinions from m different individual bloggers ($B = \{b_1, b_2, \ldots, b_m\}$). While the method is general enough, I focus on blogs to make the discussion more concrete. It is straightforward to apply our framework to other application domains. We assume that each user u_i has expressed his level of interest on each blogger b_j as the subscription score $T(u_i, b_j)$. We refer to the set of subscription scores that the user u_i places on bloggers $b_1, \ldots, b_m$

$$\vec{T_i} \quad = \quad \langle T(u_i, b_1), T(u_i, b_2), \ldots, T(u_i, b_m) \rangle$$

as the *subscription vector* of u_i. These subscription scores may be provided explicitly by the users (e.g., users may subscribe to a list of blog RSS feeds, indicating their interest in those blogs), or the subscription scores may be estimated by analyzing the past behavior of the users (e.g., by monitoring how frequently a user reads articles from each blog). I assume that there are l items of interest ($O = \{o_1, o_2, \ldots, o_l\}$) that bloggers mention and that can potentially be recommended to the users. The exact definition of an item is application dependent. For example, for a system that recommends Webpages, an item will be a Webpage; for a system that recommends electronic gadgets, an item will be an electronic gadget that is mentioned on the blogs. Whenever a blogger b_j mentions an item o_k on his blog and expresses his opinion, I assume that he provides a certain degree of "reference" for o_k, represented as the reference score $E(b_j, o_k)$. For example, when a blogger b_j includes a link to the Webpage o_k in one of his articles, we may assume that b_j is giving the reference score 1 for o_k. The set of all reference scores for the item o_k by the bloggers $b_1, \ldots, b_m$

$$\vec{E}_k \;=\; \langle E(b_1, o_k), E(b_2, o_k), \ldots, E(b_m, o_k) \rangle$$

is referred to as the *reference vector* of the item o_k.

Under this notation, the problem of computing the personalized aggregate recommendation can be stated as follows:

PROBLEM 2 For each user u_i and each item o_k, we want to compute the personalized reference score $R(u_i, o_k)$

$$R(u_i, o_k) = \sum_{j=1}^{m} T(u_i, b_j) E(b_j, o_k), \tag{4.1}$$

and return items with the highest personalized scores to the user. Equation 4.1 can be restated using the following vector notation:

$$R(u_i, o_k) \;=\; \vec{T}_i \cdot \vec{E}_k. \qquad \square$$

In order to compute the personalized reference score for each user, we have to maintain the subscription score $T(u_i, b_j)$ for every (u_i, b_j) pair and the reference score $E(b_j, o_k)$ for every o_k endorsed by b_j. These scores can be recorded in the following two tables:

- **Subscription**(user_id, blog_id, score)
- **Reference** (blog_id, item, score)

Each tuple in the Subscription table records a user's subscription score on a blogger in the score attribute. Each tuple in the Reference table records the reference score by a blogger for an item in the score attribute.

Given the above two tables, computing the items with highest personalized reference scores for user u_i can be expressed as the following SQL query $Q1$:

```
Q1:  SELECT t.item, sum(t.score*e.score) AS score
FROM Reference e, Subscription t
WHERE e.blog_id = t.blog_id AND
t.user_id = u_i
GROUP BY t.item
ORDER BY score DESC LIMIT 20
```

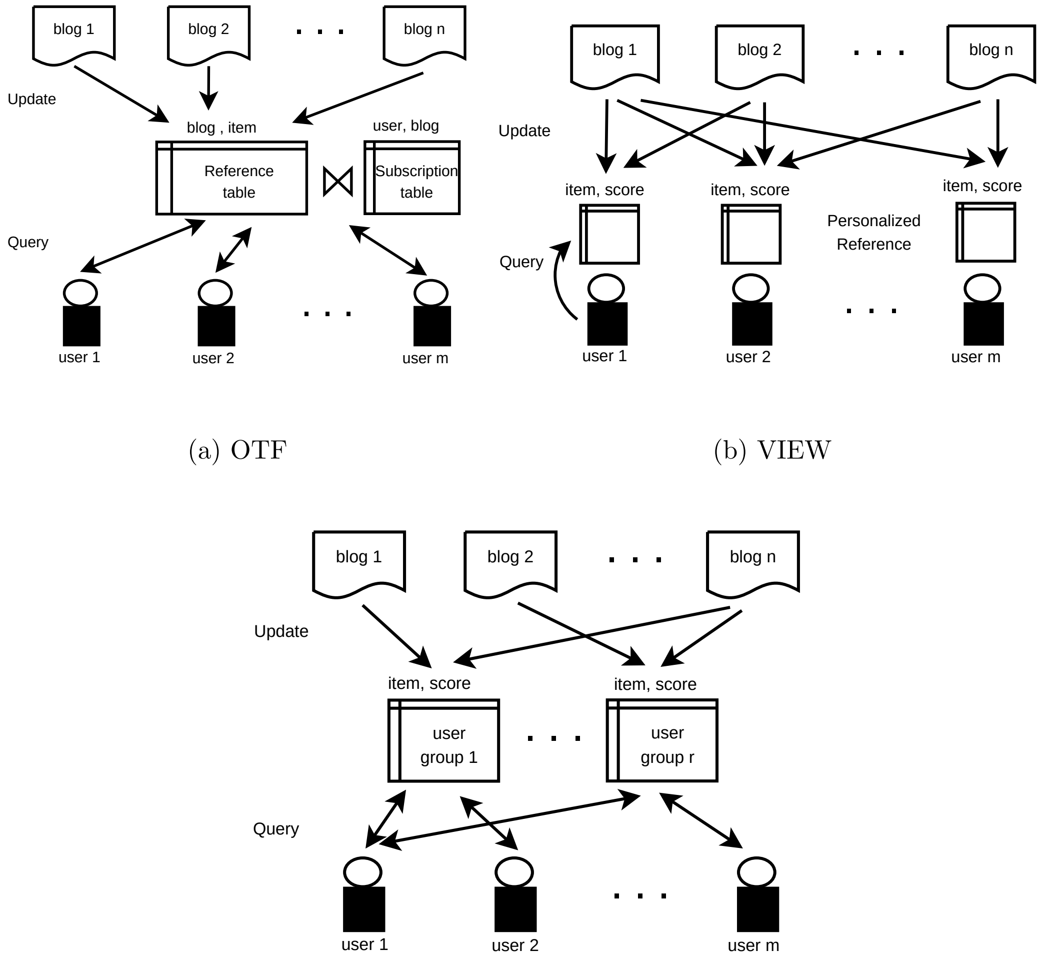
Here, I assume that we want to return the top 20 items, using the syntax *ORDER BY score DESC LIMIT 20*, for the user u_i.

4.2.1 OTF and VIEW

One simple way of returning the items with the top personalized reference scores is to compute the answer for the query $Q1$ on the fly when the user wants a personalized recommendation. This approach is illustrated in Figure 4.1(a) and is referred to as *OTF* (short for on the fly). Under OTF, one global Reference table is maintained, where a new tuple is inserted (or an existing tuple is updated) whenever a blogger posts a new article and provides a reference for an item. Using this global Reference table, we then execute $Q1$ by joining the table with a user's subscription scores in the Subscription table when the user asks for a recommendation.

Unfortunately, repeatedly executing $Q1$ for every user's request can be prohibitively expensive. Because there are millions of bloggers who keep posting new articles (and thus providing new references) and a user may subscribe a large number of bloggers and have many non-zero entries in his subscription vector, the join in Q1 may involve millions of tuples, which will be too expensive to perform on the fly.

Alternatively, to avoid this runtime query execution cost, we may proactively



(a) OTF

(b) VIEW

(c) NMF

Figure 4.1: The graphical illustration of three different methods.

precompute the personalized reference scores for every user. That is, for each user u_i, we maintain his personalized reference score table,

- **PersonalizedReference**(item, score)

as shown in Figure 4.1(b). Under this approach, whenever a blogger b_j provides a new reference for the item o_k, all PersonalizedReference tables for the users with non-zero trust on b_j are updated to reflect this new reference. This approach is referred as *VIEW*, since this approach maintains a materialized view of personalized reference scores for every user.

Under this VIEW approach, a user's personalized reference scores are always precomputed and are available in the user's PersonalizedReference table, so the user's request for a recommendation can be handled quickly. We simply need to look up the top-scored items from the user's PersonalizedReference table. However, maintaining the PersonalizedReference tables will result in significant update cost, because whenever there is a new update on a blog with a large number of subscribed users, all corresponding PersonalizedReference tables should be updated.

4.2.2 Matrix representation

In the previous section, I described the two baseline approaches for computing the personalized reference scores: OTF and VIEW. The main problem of OTF is that the query execution cost is too high due to a join between two large tables, whereas the main problem of VIEW is that the maintenance cost of the PersonalizedReference tables is too high because one new reference may trigger updates on a large number of tables. In this section, I investigate how to minimize both the table update cost and the query computation cost. In order to develop an approach with lower costs, I first reformulate the execution of query Q1 as a matrix multiplication problem.

It is easy to see that the subscription scores in the Subscription table can be viewed as an $(n \times m)$ matrix T, where each entry (i, j) represents the user u_i's subscription score on the blog b_j. Similarly, the reference scores in the Reference table can be viewed as an $(m \times l)$ matrix E, where each entry (j, k) represents the blogger b_j's reference score for the item o_k. Then from the definition of the personalized reference score $R(u_i, o_k)$ in Equation 4.1, we see that $R(u_i, o_k)$ can be computed from the matrix multiplication of T and E:

Proposition 1 *The personalized reference score of the item o_k for the user u_i, $R(u_i, o_k)$, is the (i, k) entry of the matrix multiplication of T and E. That is,*

$$R(u_i, o_k) = (TE)_{(i,k)}, \tag{4.2}$$

where $M_{(i,k)}$ represents the (i, k) entry of the matrix M. □

I use the following example to illustrate this matrix interpretation of the problem.

Example 5 Suppose that there are 3 users, 4 bloggers, and 3 items. The user u_i's subscription score on the blogger b_j is given in the subscription matrix T in Figure 4.2. For example, the value of the (1,2) entry, 0.8, indicates that u_1 subscribe b_2 with the score 0.8. Also, the blogger b_j's reference to the item o_k is given in the reference matrix E in the figure. For example, the value, 2, of the (4,2) entry indicates that the blogger b_4 refer to the item o_2 with the score 2.

Note that each row in the subscription matrix T corresponds to the subscription vector $\vec{T_i}$ of the user u_i. Also each column in the reference matrix E corresponds to the reference vector $\vec{E_k}$ of the item o_k. Given that $R(u_i, o_k) = \vec{T_i} \cdot \vec{E_k}$,

T	b_1	b_2	b_3	b_4
u_1	0.8	0.8	0	0
u_2	0.2	0.2	0.6	0.6
u_3	0	0	0.5	0.5

E	o_1	o_2	o_3
b_1	3	2	0
b_2	0	3	0
b_3	1	0	1
b_4	1	2	3

Figure 4.2: Subscription matrix T and reference matrix E.

TE	o_1	o_2	o_3
u_1	2.4	4	0
u_2	1.8	2.2	2.4
u_3	1	1	2

Figure 4.3: The result of the matrix multiplication TE.

we see that $R(u_i, o_k)$ is simply the (i, k) entry of the matrix multiplication TE. That is, computing the personalized reference score $R(u_i, o_k)$ is equivalent to performing the matrix multiplication of T and E. Figure 4.3 shows the result of this multiplication. □

Given this matrix formulation, OTF can be viewed as follows: T and E matrices are maintained separately as new reference are provided by the bloggers by updating the E matrix. Then, when the users request their personalized reference scores, *the multiplication of T and E is performed on the fly.* The cost for updating the E matrix, therefore, will be low, because a new reference from a blogger causes an update to a single entry in the matrix E; when the blogger b_j provides the reference to o_k, only the (j, k) entry of the E matrix will be updated. The computation cost of the personalized reference scores, on the contrary, will be very high because of the high cost of the multiplication of two large matrices.

The other VIEW approach can also be seen as the following matrix operations: Only the single matrix TE is maintained. New references provided by the bloggers are recorded by updating the appropriate entries in TE. Since all personalized scores $R(u_i, o_k)$ are precomputed in the matrix TE and are readily available, the cost for answering a user's request for $R(u_i, o_k)$ will be low. By contrast, the cost for maintaining the matrix TE will be high, because a single reference from a blogger may trigger multiple updates on a large number of entries in the TE matrix. For example, a reference from b_j who has 1,000 subscribed users will incur updates to 1,000 entries in the TE matrix.

The above two extreme approaches, either precomputing the entire matrix multiplication TE or performing it lazily at the user's request, suggest exploring a possible middle ground, where part of the multiplication is performed proactively before the user's request and the rest of the multiplication is finished on the fly. In the next section, we will see that finding this middle ground is equivalent to finding a good low-rank decomposition of the matrix T.

4.2.3 Matrix decomposition for efficient computation

To understand how matrix decomposition can be used to find the middle ground, Let us consider the subscription matrix T in Figure 4.2 of Example 5. From the Subscription table, it is observed that even though there are 3 users with different subscription vectors, the user u_2's vector $\vec{T_2}$ is simply a linear combination of $\vec{T_1}$ and $\vec{T_3}$. That is,

$$\vec{T_2} = 0.25\,\vec{T_1} + 1.2\,\vec{T_3}. \tag{4.3}$$

Now, given that $R(u_2, o_k) = \vec{T_2} \cdot \vec{E_k}$, it is observed that $R(u_2, o_k)$ can actually be computed from $R(u_1, o_k)$ and $R(u_3, o_k)$. That is,

$$R(u_2, o_k) \;=\; \vec{T_2} \cdot \vec{E_k}$$

$$
\begin{aligned}
&= (0.25\,\vec{T_1} + 1.2\,\vec{T_3}) \cdot \vec{E_k} \\
&= 0.25\vec{T_1} \cdot \vec{E_k} + 1.2\vec{T_3} \cdot \vec{E_k} \\
&= 0.25R(u_1, o_k) + 1.2R(u_3, o_k).
\end{aligned}
$$

This observation hints at a possible modification to the VIEW approach: *instead of maintaining one PersonalizedReference table for every user, maintain the table only for the two "representative users," u_1 and u_3. The personalized reference score for u_2 is then computed indirectly by combining the personalized scores for u_1 and u_3.* This modification to the VIEW approach has the following two merits in terms of its update and the query computation cost:

1. *Update cost*: The update cost for the PersonalizedReference tables is significantly reduced from the VIEW approach. That is, according to the subscription matrix T in Figure 4.2, every blogger b_j is subscribed by two users (i.e., for every column b_j in T, there are 2 non-zero entries). Therefore, under the VIEW approach, a new reference made by the blogger b_j will trigger updates to *two* Reference tables. In contrast, under the modified approach, we maintain only two PersonalizedReference tables for u_1 and u_3. Now, because b_j is subscribed by only one of u_1 and u_3, a new reference made by by b_j will trigger just a single update to one of the two PersonalizedReference tables.

2. *Query computation cost*: The computation cost for personalized reference scores of the modified approach is also significantly lower than that of OTF. Under OTF, the computation of $R(u_i, o_k)$ values involves the multiplication of T and E matrices. Under the modified approach, by contrast, $R(u_i, o_k)$ scores for the user u_1 and u_3 have already been precomputed. For the remaining user u_2, the computation of $R(u_2, o_k)$ can also be done cheaply

by taking the weighted sum of $R(u_1, o_k)$ and $R(u_3, o_k)$. In short, the query computation cost of the modified approach is significantly lower than that of OTF and is close to the cost of VIEW.

The modified approach to computing $R(u_i, o_k)$ can be viewed as the following matrix decomposition of T into W and H:

$$\begin{aligned} T &= WH \\ \begin{pmatrix} 0.8 & 0.8 & 0 & 0 \\ 0.2 & 0.2 & 0.6 & 0.6 \\ 0 & 0 & 0.5 & 0.5 \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ 0.25 & 1.2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} .8 & .8 & 0 & 0 \\ 0 & 0 & .5 & .5 \end{pmatrix}. \end{aligned}$$

Here, each row of the second matrix H corresponds to the subscription vector of the two "representative" users, u_1 and u_3. Each row of the first matrix W then represents how each user u_i's subscription vector can be obtained from the subscription vectors of the two representative users. For example, the first row of W is 1 for the first entry, indicating that u_1's subscription vector is identical to the subscription vector of the first representative user. The second row has 0.25 and 1.2 as its entries, indicating that the subscription vector of u_2 is the the linear combination of the subscription vectors of the two representative users, weighted by 0.25 and 1.2, respectively.

Given this decomposition, we see that maintaining the PersonalizedReference tables for the two representative users u_1 and u_3 is equivalent to precomputing the multiplication of H and E. Note that each row of the HE matrix has the $R(u_i, o_k)$ scores of each of the two representative users. Then during the query time, $R(u_i, o_k)$ of u_i is computed by multiplying the precomputed HE with W as follows:

$$
\begin{aligned}
TE &= (WH)E = W(HE) \\
&= \begin{pmatrix} 1 & 0 \\ 0.25 & 1.2 \\ 0 & 1 \end{pmatrix} \left(\begin{pmatrix} 0.8 & 0.8 & 0 & 0 \\ 0 & 0 & .5 & .5 \end{pmatrix} \begin{pmatrix} 3 & 2 & 0 \\ 0 & 3 & 0 \\ 1 & 0 & 1 \\ 1 & 2 & 3 \end{pmatrix} \right) \\
&= \begin{pmatrix} 1 & 0 \\ 0.25 & 1.2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2.4 & 4 & 0 \\ 1 & 1 & 2 \end{pmatrix}.
\end{aligned}
$$

The main benefit of this approach comes from the fact that the number of rows in H is much smaller than the number of rows in T, so a much smaller number of PersonalizedReference tables are needed. More formally, both the update and the query computation costs for personalized reference scores are reduced by decomposing the $(n \times m)$ matrix T into $(n \times r)$ and $(r \times m)$ matrices W and H, where $r \ll n$, and by precomputing the multiplication HE.

In general, this process of matrix decomposition has the following intuitive interpretation: The *representative groups of users* who have similar subscription vectors are first identified so that we can precompute the personalized reference scores for each representative user group. Then during the query time, we compute the personalized reference scores of each user by combining his scores on the representative user groups. Figure 4.1(c) graphically illustrates this interpretation. Note that, in the figure, we maintain *one PersonalizedReference table per user group*, not per user. Since the number of user groups is much smaller than the number of users, this approach is likely to lead to significant improvement in the update and the query computation cost compared to VIEW and OTF.

4.2.4 SVD and NMF

The previous discussion shows that we can reduce the costs of multiplication by finding a way to decompose $(n \times m)$ matrix T into $(n \times r)$ matrix W and $(r \times m)$ matrix H where $r \ll min\{n, m\}$. Since r corresponds to the number of "representative" user groups for which we maintain separate PersonalizedReference tables, one would like to find a decomposition WH such that r is minimal. Unfortunately, it is known that the minimum r value for the decomposition WH is the *rank* of the matrix T. That is, for a matrix T whose rank is close to $min\{n, m\}$, it is not possible to obtain a decomposition WH with a low r value. Given that exact decomposition is almost impossible, we relax our goal to find the closest approximation of the subscription matrix T with WH for a given r value.

Desiderata 1 Given the desired rank r of the matrix decomposition of T, find the W and H such that WH is closest to T:

$$T \approx WH. \qquad \square$$

One well-known method for low-rank matrix approximation is Singular Value Decomposition (SVD). If SVD is used to decompose matrix T into two orthogonal matrices, the decomposition is guaranteed to be the best in terms of the Forbenius norm, at any given rank approximation. Unfortunately, the decomposed matrices W and H result in significant update and query execution cost due to the high density of the matrices; almost all entries in the W and H are non-zero, which can be interpreted to mean that every blogger b_j is subscribed by every representative user group and that every user u_i belongs to every representative user group to a certain extent. Therefore, an update from blogger b_j results in updates to the PersonalizedReference table of almost *every* user group. Also, the computation of $R(u_i, o_k)$ for the user u_i requires us to combine the scores from almost *all*

PersonalizedReference tables.

The high update and query costs resulting from the SVD decomposition led us to introduce the second desiderata for the matrix decomposition:

Desiderata 2 Given the desired rank r of the matrix decomposition of T, find the W and H, $T \approx WH$, such that matrices W and H are as sparse as possible. □

Intuitively, by making most of the entries of W and H zero, it is trying to identify the user groups such that each blog b_j is subscribed by only a small number of groups and each user u_i belongs to only a small number of groups.

Non-negative Matrix Factorization (NMF) is one of the methods with both properties. In fact, NMF allows us to specify the desired sparsity of the resulting matrix W and H for a given rank value r. Later in Section 4.3, I will provide the experimental comparison between SVD and NMF, in terms of their approximation accuracy and the sparsity of the decomposed matrices. In the rest of this chapter, I refer to the approach of using NMF for the matrix decomposition and the $R(u_i, o_k)$ computation as the *NMF method.*

4.2.5 Efficient computation of top K items

Once the PersonalizedReference tables are precomputed for every user group using the NMF method, we can compute $R(u_i, o_k)$ for u_i by combining the scores from the tables u_i belongs to. In most cases, as the query Q1 suggests, users are interested in top K items with the highest personalized reference scores instead of every item referenced by bloggers. This suggests another possibility for optimization under the NMF method. Instead of computing $R(u_i, o_k)$ for every item o_k, we compute $R(u_i, o_k)$ only for the items that are likely to have high scores. More specifically, from each PersonalizedReference table that u_i belongs to, we obtain only those items with high $R(u_i, o_k)$ scores in each table. We then compute the personalized $R(u_i, o_k)$ values only among these items and return the top K. This way, we can further reduce the execution cost of $Q1$ because only a few tuples from each PersonalizedReference table are read instead of scanning most of the tuples in the tables.

In [FLN01], Fagin *et al.* proposed an algorithm, called Threshold Algorithm, suitable for this optimization. Through a careful analysis of the algorithm, the authors have shown that we can indeed compute the *correct* top K items optimally, by looking at just the top few entries from each PersonalizedReference table. More formally, they have shown that when the score of an item o_i is computed by a monotone aggregate function $t(o_i) = t(x_1, \ldots, x_m)$, where $x_1, \ldots, x_m$ are the basis scores from which the final score is computed, we only have to read the top items with the highest $x_1, \ldots, x_m$ scores until a certain threshold condition is met. See [FLN01] for more detailed description of the algorithm. Later in the experiment section, I implement Threshold Algorithm for the top K item computation and report the performance numbers with this optimization. Note that the optimization based on the Threshold Algorithm does not involve any approximation, because the returned top K items are guaranteed to be the correct top K items.

4.2.6 Hybrid approach

The NMF method brings advantages by approximating and decomposing a dense user-blog subscription matrix into fewer numbers of user groups; hence, there is no need to update a large number of PersonalizedReference tables or to aggregate a large number of tuples in the Reference table at query time. However, the user-blog subscription matrix may not be dense over all users and blogs. In a real-world dataset, some users may only subscribe to a few blogs, while some blogs may only have a few readers. For these kinds of users and blogs, the two baselines, OTF and VIEW, can already handle the personal aggregate query efficiently.

As shown in the literature [JKF07] (and also in the experiments afterwards), the user-blog subscription matrix is usually skewed and shows a power-law-like distribution. Figure 4.5 shows a subscription matrix with rows and columns as the blogs and users respectively. When the rows and columns of the matrix are ordered by the number of non-zero entries, it may be divided roughly into three regions, as illustrated in Figure 4.4. In the figure, the region marked as "1. OTF" corresponds to the users that subscribe to just a small number of blogs. The other two regions, "2. VIEW" and "3. NMF," correspond to the users who subscribe to a large number of blogs, where "2. VIEW" indicates the blogs with just a few subscribers and "3. NMF" indicates the blogs with many subscribers.

Based on such division of users and blogs, different methods can be applied to process aggregate queries. For the region "1. OTF," OTF is efficient enough; since the users in the region subscribe to a small number of blogs, personalized reference score is computed from just a small number of blogs. For the other two regions, the query computation cost can be high if OTF is used, since these users subscribe to a large number of blogs. Fortunately, for the region "2. VIEW," the VIEW method can be used to precompute the PersonalizedReference table for each user whenever there is a new reference made by each blog; the blogs in this region are subscribed by a small number of users, so an update from a blog in this region triggers updates to just a few PersonalizedReference tables. The third region, "3. NMF," however, results in too much cost if either OTF or VIEW method is used, so we apply the NMF method to this region to reduce the

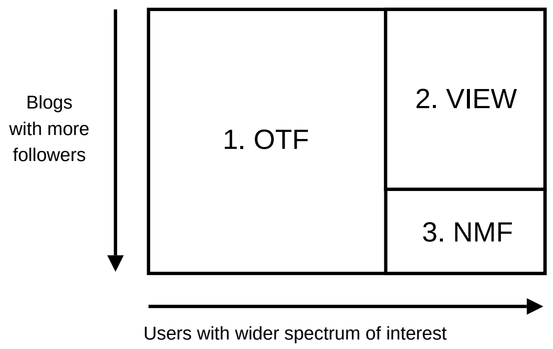

Blogs with more followers
1. OTF
2. VIEW
3. NMF
Users with wider spectrum of interest

Figure 4.4: Three different regions of the user-blog subscription matrix.

query computation cost. Later, in Section 4.3, we experimentally investigate how the relative size of the three regions impacts the query and update performance of the overall system.

Under such a partitioning scheme, whenever new users or blogs are added, they can first be handled in region “1.OTF” and “2.VIEW” because they are assumed to have a smaller number of subscriptions and subscribers respectively, for which the query cost and update cost are low. Either periodically or after the subscription data has changed from the previous version beyond a threshold, the subscription matrix is repartitioned and the region “3.NMF” is recomputed again to update these changes. In the proposed model, I assume that the subscription matrix is fairly stable and will leave the study of handling a dynamic subscription matrix as future work.

4.3 Experiments

In this section, I evaluate the effectiveness of the proposed NMF method. In Section 4.3.1, I first describe the dataset used for the experiments. Then in Section 4.3.2, I investigate how much difference personalization makes in the recommendation quality by comparing recommended items with or without personalization. In Sections 4.3.3 and 4.3.4, I evaluate the effectiveness of the NMF method by measuring the accuracy of the NMF method in approximating the top k popular items (Section 4.3.3) and by comparing the query and update cost of the NMF method to the other two baseline methods (Section 4.3.4). Finally, in Section 4.3.5, I measure how different choices of parameters affect the performance of the NMF method.

4.3.1 Description of dataset

Subscription matrix The users' subscription information on the blogs is obtained by collecting a snapshot of the user-blog subscription data from Bloglines [Blob]. Bloglines is a Web-based online RSS reader, where users can specify the list of RSS feeds that they are interested in so that they can access new articles from the subscribed blogs at a single location.

This subscription dataset contained 91,366 users who subscribed to 487,694 distinct RSS feeds on the Web.[3] On average, one user subscribed to 30 distinct RSS feeds, leading to a total of 2.7 million user-blog subscription pairs. Figure 4.5 shows the collected subscription matrix, where the users and the feeds are sorted by their number of subscriptions and subscribers, respectively. The subscription pattern follows a power-law distribution as reported in similar previous

[3] Only users with public profiles are considered.

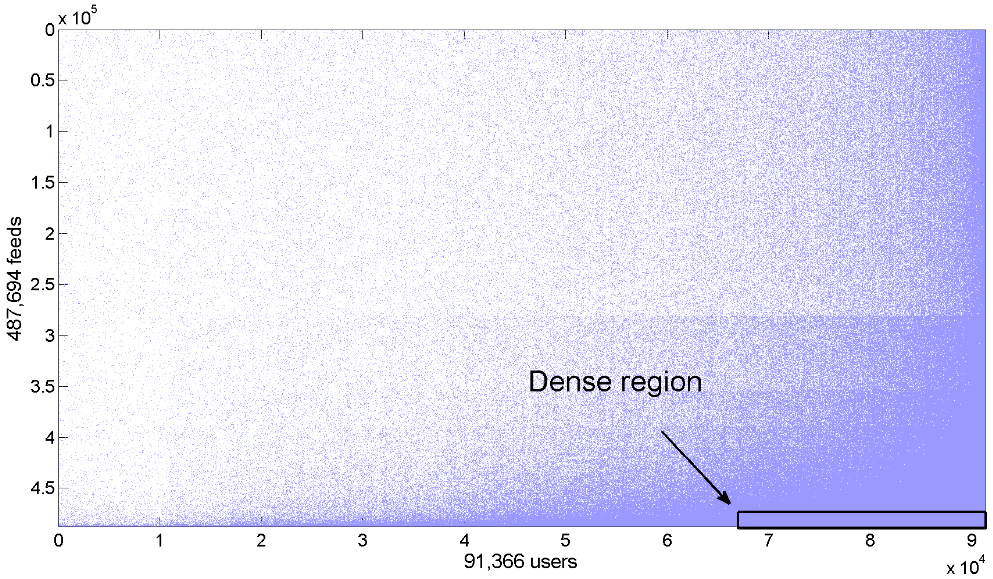


Figure 4.5: Subscription matrix with rows and columns ordered by the number of subscribers and subscriptions, respectively.

studies [JKF07]. The matrix is indeed very sparse, with most of the subscription pairs located in the lower right-hand corner. (This may not be noticeable because of the printing difficulty) There were 24,340 users with more than 30 subscriptions and 10,152 blogs with more than 30 subscribers, which corresponds to the box in the lower right corner in Figure 4.5. This region consists of roughly 1 million subscription pairs. Since the Bloglines users do not indicate their levels of interest over the feeds, the subscription matrix is a degenerate version of the user-blog subscription matrix having values of either zero or one.

Reference matrix In order to obtain the reference matrix for the experiments, all articles posted at the 487,694 RSS feeds are collected between October 2006 and July 2007 and their contents are analyzed. Again, we consider that when the item o_i appears in an article posted by the blogger b_j, the blogger "reference" the item. Still, an important decision that we have to make is what constitutes an item. For the experiments, I explored two possibilities – the *URLs* appearing on blog articles and the *keywords* appearing on the blog articles, only results obtained from keywords are shown in this chapter. This choice of item may be interpreted as identifying "popular buzzwords" within the subscribed blogs for each user. To assign the reference score for every keyword appearing in the blog articles, we extracted only the nouns from the articles, using a basic NLP part-of-speech tagger and used the standard $tf.idf$ score of those nouns, where *idf* is calculated from all the blog entries published on the same day. There are other methods like n-grams technique and KL-divergence with background corpus [GHT04], to extract key phrases from blog articles to improve the quality of recommendation; I apply the $tf.idf$ method for simplicity and focus my work on the query optimization.

4.3.2 Does personalization make a difference?

To quantitatively show how much difference personalization makes in the overall recommendation, I first present the top few recommended items when they are computed *globally* by aggregating the reference scoress from all bloggers with equal weights and when they are computed *individually* for each user by weighting the reference scores with each user's subscription vector. Table 4.1 shows the list of top 10 recommended keywords among all RSS feeds and for three sample users in the week between 2007-01-07 and 2007-01-13. This is the week when Apple Inc. announced its iphone.

From this list we can see that personalizing the aggregation based on a user's interest does make a big difference in terms of the recommendation. Globally, the announcement of iphone by Apple was indeed very popular among bloggers and showed up as one of the top 10 recommended keywords. This globally popular event, however, was essentially filtered out for the recommendations for the user 90550 and the user 91017, whose interests seem to be less technology-oriented, but more towards media/entertainment (user 90550) and politics (user 91017). We also show the same list computed in the week between 2007-04-01 and 2007-04-07. The global top keywords, again, pick up an important event happened in the week (the *Easter holiday*), while top keywords for individual users are less sensitive to the change of the global trend and continue to be related to their personal interests.

To quantitatively measure the difference of the recommended keywords among the users, I compute the following metrics. Let L_G be the set of global top 20 keywords and L_i be the set of top 20 keywords of the user i. Then the average overlap between the global recommendation and the individual recommendations can be measured by $\frac{1}{n}\sum|L_G \cap L_i|$, and the overlap of the recommendations among the users can be measured by $\frac{2}{n(n-1)}\sum_{i \neq j}|L_i \cap L_j|$, where n is the number of users. When we measure these overlaps, they were 1.12 and 1.13, respectively, for the top 1000 users with the largest number of subscriptions, indicating that the recommended keyword lists shared only one keyword, on average.[4]

The shared-keyword count distributions in shown in Figure 4.6(a) and 4.6(b). As we can see, even among users with a large number of subscriptions (that are likely to have overlapping interests), their personalized answers differ significantly from the global answer and also among the users themselves.

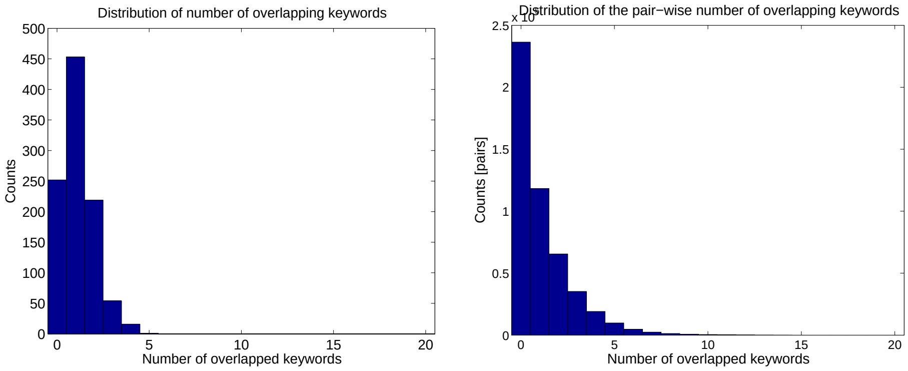
4.3.3 Accuracy of approximation

The NMF method attempts to compute the aggregation quickly, at the cost of losing the accuracy of the aggregation. In this section, I investigate the accuracy

[4] These numbers were even smaller for the users with fewer subscriptions.

Global	user 90439	user 90550	user 91017
2007-01-07 to 2007-01-13			
sales	cattle	brazil	yorker
iphone	beef	iguazu	iraq
apple	iphone	reuters	bush
manager	chicago	search	president
iraq	iraq	vegas	views
management	bush	argentina	avenue
development	apple	kibbutz	dept
software	companies	video	troops
business	prices	cathartik	saddam
phone	quarter	google	iran
2007-04-01 to 2007-04-07			
easter	bush	angeles	yorker
news	iraq	google	views
google	campaign	kibbutz	rock
sales	president	premiere	theatre
business	slashdottit	entourage	critic
description	money	photo/gus	iran
york	chicago	ruelas	paul
police	plans	shop	fiction
security	zell	reuters	southern
quality	mccain	actress	stage

Table 4.1: Global and individual lists of top keywords during the weeks of 2007-01-07 to 2007-01-13 and 2007-04-01 to 2007-04-07.



(a) Compare with global

(b) Pair-wise

Figure 4.6: Distribution of the number of overlapping top 20 keywords among top 1000 users.

of the approximated result computed by the NMF method.

I first compare NMF with SVD on how close they can approximate the original matrix when both are approximated to the same rank.[5] Since NMF does not necessarily produce orthogonal matrices as the output, when the SVD approximation is done to rank-r, I choose $n \times r$ and $r \times m$ to be the size of W and H used in NMF, for a fair comparison with the r rank SVD approximation. Table 4.2 reports the accuracy of these two methods. The first column shows the rank of the approximated matrix. The second column shows the Frobenius norm of the difference between the original matrix and the SVD-approximated matrix (i.e., $|T - USV^T|$), and the third column shows the Frobenius norm under the NMF approximation (i.e., $|T - WH|$). From this result, we can see that SVD and NMF both result in roughly the same accuracy in terms of the Frobenius norm; the NMF approximation is only 1% worse than that of the SVD; however,

[5] SVD is proven to provide the best approximation under the Frobenius norm for a given rank.

	SVD	NMF
rank	norm	norm (sparsity of W, H)
80	848.5	856.9 (25.0%,14.4%)
90	841.6	850.1 (24.7%,13.6%)
100	835.1	844.6 (23.2%,13.3%)
110	829.0	837.9 (22.7%,12.9%)
120	823.2	833.0 (21.5%,12.7%)

Table 4.2: Comparison of the accuracy and sparsity between SVD and NMF at different rank approximation.

NMF significantly outperforms SVD in terms of the sparsity (the percentage of non-zero entries) of the decomposed matrices. From the third column of the table, we see that NMF gives an average sparsity of 23% in W and 13% in H, while the sub-matrices decomposed from SVD contain almost 100% non-zero entries.

To make the accuracy of the approximation easier to see visually, Figure 4.7 shows the density map of the subscription matrix (only the dense region) comparing the original subscription matrix, the SVD, and the NMF approximations. From the figure, we can see that both NMF and SVD leads to a very close approximation of the original subscription matrix.

In addition to measuring the accuracy in terms of the subscription matrix approximation, I investigate NMF's accuracy in terms of the top 20 recommended items to the users. To quantify this, suppose A_i and T_i are the top k recommended items computed with and without using the NMF approximation method for the user u_i. Then the degree of overlap between the correct and approximated recommendations, $\frac{A_i \cap T_i}{T_i}$, is measured. It is observed that, averaging among the top 1000 users (those with the largest number of subscriptions), 70% of the

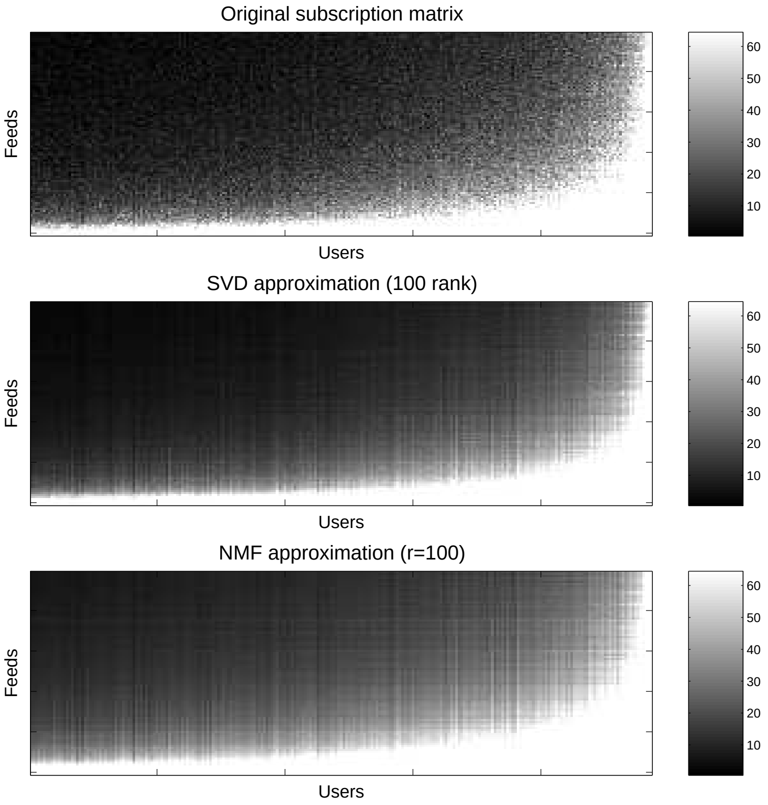


Figure 4.7: Visual comparison of the accuracy of subscription matrix approximation.

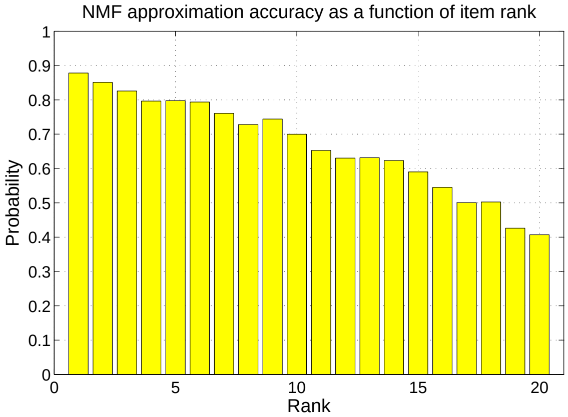


Figure 4.8: Approximation accuracy as a function of rank.

recommended items were shared between the two lists. Figure 4.8 shows that this overlap varies as we change k in the top k. From the figure, we can see that the higher-ranked items, which are considered more important to the users, are more likely to be approximated by the NMF method. For example, the top-ranked item was recommended by the NMF method in about 90% of the cases.

4.3.4 Efficiency of the NMF method

I now compare the update and query performance of NMF with two other baselines, OTF and VIEW. To compare their update performance, I assume that we monitor the blogs for one week from 2007-01-07 to 2007-01-13, and as new items appear in the blogs, we update the Reference table in OTF, the PersonalizedReference table of users in VIEW, and the group tables in NMF. From this measurement, we see that the total numbers of updates for OTF, VIEW, and NMF are roughly 222K, 23.6M, and 3.2M, respectively, for the region where the NMF approximation is applied. That is, while the update cost of NMF is higher than that of OTF, NMF still reduces the update cost by an order of magnitude

Method	avg	std	max	min
OTF	2.05s	3.60s	84.42s	0.037s
NMF	0.46s	0.53s	2.84s	0.007s

Table 4.3: Statistics of time taken by OTF and NMF (among the top 1000 users).

compared to VIEW.

Table 4.3 reports the query response time of OTF and NMF for top 1000 users (with a large number of subscriptions). The result from VIEW is not reported here, because answering a query under VIEW is a simple table lookup. The response time of OTF is measured by running the $Q1$ in Section 4.2 on MySQL, and that of NMF is measured by a python implementation of the NMF and Threshold Algorithm interfacing a MySQL 5.0.27 server (with a 600MB main memory as index key cache) running on an AMD Dual Core Fedora machine with database files residing on a RAID disk. On average, the OTF method spends 2.05s to answer a query, while the NMF method spends 0.46s. The table shows that the reduction in the query response time is even more significant when we compare the maximum response time; NMF reduces the maximum from 84.42s to 2.84s and allows all users to get results within a reasonable amount of time in an interactive setting.

In summary, we can see the NMF method is a middle ground between OTF and VIEW in trading more update cost for a faster query response time.

4.3.5 Sensitivity analysis of NMF region size

I have previously studied using an arbitrary choice (users with >30 subscriptions and feeds with >30 subscribers) of the dense subscription region to apply the

boundary	size	# subscription pairs
(feeds, users)	$(m \times n)$	(sparsity)
$> 30, > 25$	$10,152 \times 28,236$	1,070,130 (0.37%)
$> 30, > 30$	$10,152 \times 24,340$	1,004,908 (0.41%)
$> 30, > 35$	$10,152 \times 21,182$	944,385 (0.44%)
$> 30, > 40$	$10,152 \times 18,600$	885,581 (0.47%)
$> 30, > 45$	$10,152 \times 16,392$	834,423 (0.50%)
$> 25, > 30$	$12,470 \times 24,340$	1,060,292 (0.35%)
$> 30, > 30$	$10.152 \times 24,340$	1,004,908 (0.41%)
$> 35, > 30$	$8,514 \times 24,340$	958,789 (0.46%)
$> 40, > 30$	$7,275 \times 24,340$	918,483 (0.52%)
$> 45, > 30$	$6,315 \times 24,340$	883,324 (0.57%)

Table 4.4: Characteristics of different sizes of NMF region.

NMF method. In this experiment, I empirically try different sizes of the NMF region and study its impact on the query cost, update cost, and approximation accuracy. Table 4.4 shows the size and sparsity of the NMF region under different choices of boundary. For each of these settings, the NMF method is applied with the same parameter $r = 100$.

Figure 4.9 shows how the update cost (as the number of SQL update statements used to process one week of data) changes with the size of the NMF region. It is observed that changing the NMF region size along the users and feeds dimension has an opposite effect on the update cost. The blue solid line shows that when fewer users are included in the NMF region (i.e., more users are handled by the OTF method), the update cost is lower because users handled by the OTF method do not require their PersonalizedReference table to be maintained. The

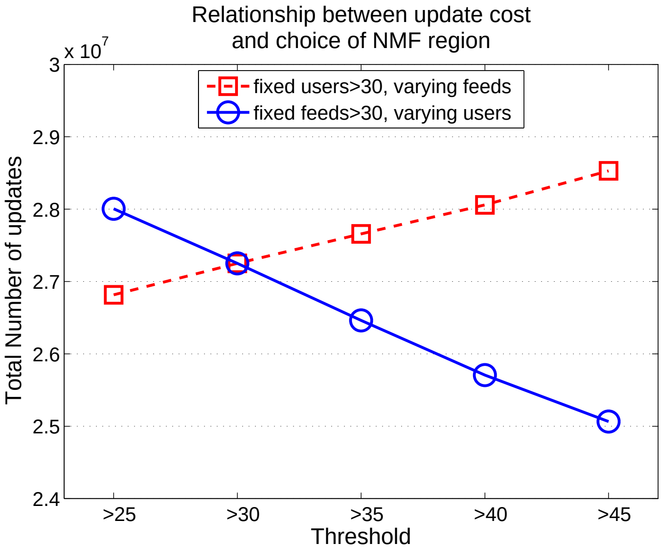


Figure 4.9: The impact of NMF region size on update cost.

red dashed line shows that when fewer feeds are included in the NMF region (i.e., more feeds are handled by the VIEW method), the update cost is higher because fewer feeds are benefited from the NMF method to reduce the number of updates. This relationship suggests that we ought to include fewer users but more feeds to be approximated by the NMF if update cost minimization is the desirable objective.

Next, I examine the impact of the size of the NMF region on the query efficiency and accuracy of approximation. Since the evaluation focuses on the set of users with the largest number of subscriptions, changing the NMF boundary along the user dimension will not affect the group of users evaluated; therefore, I investigate only the change along the feed dimension that determines the proportion of feeds to be handled by VIEW and the NMF method.

Figure 4.10 shows the average approximation accuracy (with ±1 standard deviation) of the top 1000 users under different sizes of the NMF region. It is

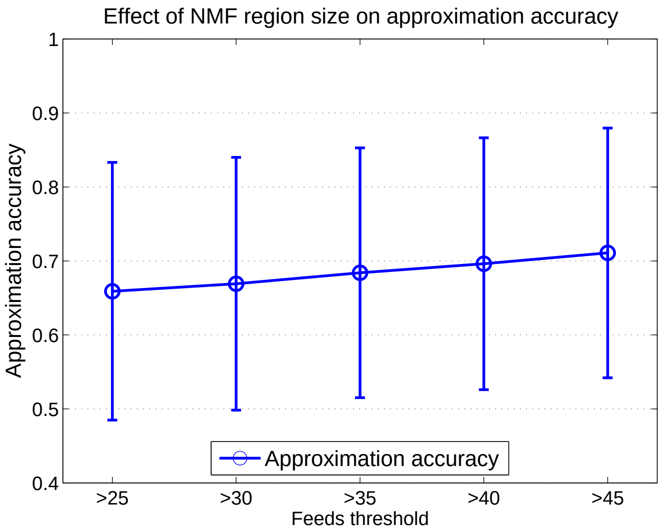


Figure 4.10: The impact of different sizes of NMF region on approximation accuracy.

observed that when fewer feeds are being approximated by the NMF method, the approximation accuracy is higher because more feeds are now handled by the VIEW method without any approximation. Other than this, it shows that the query response time remains similar regardless of the NMF region size. This can be explained by the fact that the Threshold Algorithm is combining from a similar number of ordered lists because the sparsity of the factorized matrices is similar when the same r parameter is used for different NMF region sizes.

Based on the above observation, when we want to best leverage the advantage brought by the NMF method, we should aim at including fewer users but more feeds in the NMF region so that we can reduce the update cost while still giving good approximation accuracy.

4.4 Related Work

The work presented in this chapter spans over several areas such as Web data mining, personalization, and query optimization, the related literature falls roughly into three categories: 1) Web mining for trends, 2) Web personalization and collaborative filtering, and 3) OLAP and query optimization of aggregates and ranking operations.

Blogging activities have brought to the Web a huge amount of user-generated content. There have been research efforts [GHT04, GGL04, MLS06, WZH07] on mining Web data for trends and information retrieval specific to blog data. For example, Gruhl *et al.* [GGL04] describe several techniques to find representative keywords and phrases (memes) that correspond to discussion topics in the blogosphere and present methods to trace the spreading of information among blogs. Wang *et al.* [WZH07] have proposed a method to correlate similar trends from multiple sources. The majority of the prior work on Web mining focuses on finding a set of global trends by aggregating a large number of sources, while my work further extends this idea to provide more personalization to improve user experience.

There is a significant amount of work on Web personalization and collaborative filtering, ranging from the fast learning of accurate user profiles [QC06] to personalized recommendation through collaborative filtering [PP07] to the optimization of complex systems [KI05, DDG07]. Recently, Das *et al.* [DDG07] describe how *Google News* provides personalized news feed items. It addresses the implementation of PLSI and EM algorithm using the map-reduce programming paradigm and user profile management within the Google cluster. We see that applying the NMF method in a distributed environment to further improve efficiency will be an interesting investigation.

Optimization of aggregation and ranking query has long been studied in the database community, especially in the OLAP context. Recently, Li *et al.* [LWL07] have introduced an extension to the SQL semantics to support the custom clustering method using “group by” and to support the general ranking function using “order by.” Qu and Labrinids [QL07] describe how to optimize the scheduling process of a streaming database engine to accommodate different user preferences for freshness and accuracy of results. While some other previous work [QQZ06, KWF06, LCI06] on efficient aggregate query processing may differ from application domains and assumptions, they share the same line of thought to improve query processing through approximation and reusing partially computed results among queries.

My work benefits from two existing prior arts: 1)The core technique, Non-negative Matrix Factorization, had been applied by machine learning researchers in various application domains such as pattern recognition [Lin07], computer vision [Hoy04], and clustering [XLG03, DLP06]. 2) The Threshold Algorithm is an efficient method proposed by Fagin *et al.* [FLN01] to rank objects by merging from multiple sorted attribute lists.

4.5 Summary

In this chapter, I formalized the problem of personalized aggregation. I presented two baseline approaches and discussed their limitations. I then presented a matrix representation of the problem and proposed a method that uses Non-negative Matrix Factorization and Threshold Algorithm to speed up the query processing and reduce the update cost. Using experiments on a real-life blog dataset, I showed that the personalized aggregation is able to provide different results among users. I also demonstrated the effectiveness of my proposed solution through an analysis of update cost based the bloglines dataset and a prototype implementation on a commercial database to evaluate query cost. In particular, the NMF method is able to cut the query response time by 75%, at the expense of paying a reasonable amount of update cost, while maintaining an approximation accuracy of roughly 70%. Some interesting future directions include investigating the optimal choice of the number of users and blogs to be included in the NMF approximation, the effect of matrix sparsity on query efficiency, and the application of the NMF method in a distributed computing environment.

CHAPTER 5

Social annotation analysis

5.1 Introduction

The online content aggregator collects a lot of user activity data over time, for example: What materials are marked as her favorite? What keywords does she use as tags? These data not only serve as the basis for providing personalized recommendations but also provide information about Web-resource categorization and vocabulary usage in the information-retrieval process. Social annotations, such as bookmark tags, video annotations, and query logs of search engines, are being generated both implicitly and explicitly every day. Such data involve intensive human effort to match the best set of descriptive keywords with their corresponding Web resources. The word-usage patterns learned from such social annotations can then be transferred to improve other online applications, for instance, online advertising keyword selection.

Search engine's sponsored search services, such as Microsoft's adCenter [Mic] and Google's Adwords [Gooa], provide advertisers a convenient way to reach a potentially much larger and highly targeted audience as compared to traditional media advertising. Moreover, such search engines make it easier to measure the effectiveness of the advertisement. These factors all contribute to the emergence of online sponsored search markets, with an estimate of over 17 billion dollars spent in 2006 at a growth rate of 20% [AFJ07]. A simplified but typical sponsored search activity requires advertisers to design an advertisement snippet and bid on a set of selected keywords, while search engines select and show the relevant snippets [1] whose bidding keywords match users' queries. Besides relying on search engines to match relevant queries to advertisements, advertisers can play an active role by identifying a good set of keywords to improve the effectiveness. Selecting a good set of advertising keywords, however, is never an easy problem and requires considerable human effort. For example, a large retail store may have to identify and choose thousands of keywords manually to bid in an advertising campaign, while a typical online store might be looking for automatic ways to analyze its Website to mine for possible advertising keywords. These factors drive us to apply social annotations for selecting effective advertising keywords.

An advertiser would like to maximize his benefit by bidding the best set of keywords to catch user attention and attract relevant traffic to his Website. For example, some of the questions he has in mind will be: 1) Is the word specific enough to ensure that my advertisements will be shown to my target audience? 2) Does the word correspond to some emerging topics in which users are more willing to explore new pages, such as the one on which I am advertising? 3) Will the word drift its meaning and topic association over time, thus lowering its effectiveness in bringing relevant traffic to my Website? How can we use social annotations to detect keywords with these properties?

In this chapter, I present methods that intend to help advertisers select effective advertising keywords based on social annotations collected by an online content aggregator by making the following contributions:

- I study a social annotation dataset, *delicious* [Del], and apply a state-of-the-art text-mining algorithm to demonstrate its power to categorize both

[1] Search engines may also consider the CTR (click through rate) of advertisement snippets as a form of relevance in order to maximize their income.

Web resources and word-usage behavior in Section 5.2.

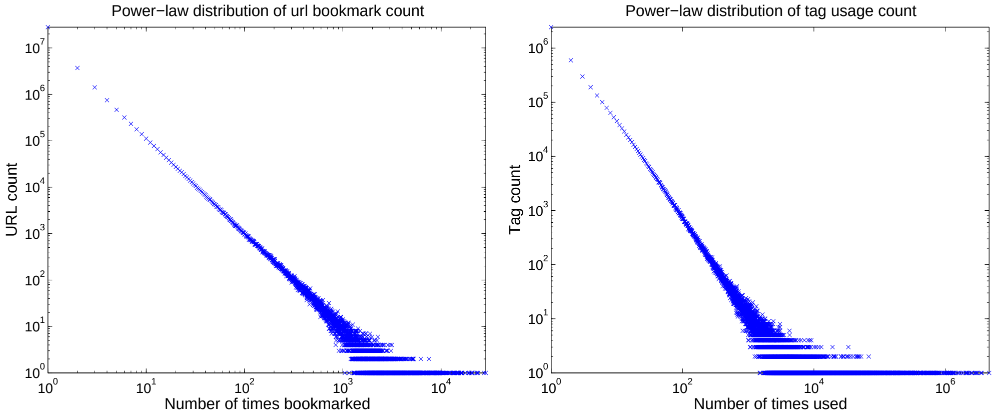
- I identify three desiderata of advertising keywords selection that are favorable to online advertisers and explore the corresponding features in the social annotation data to characterize them in Section 5.3.

- I conduct an experiment, presented in Section 5.4, to solicit users' judgment on the three desiderata of advertising keywords. The ground truth obtained is used to build an advertising keywords classifier with good accuracy. A correlation analysis also confirms that the appropriate features are chosen to characterize effective advertising keywords.

5.2 Social annotations

Social annotations involve considerable human effort to associate the best set of descriptive words with their corresponding Web resources over time, such as annotations of videos, bookmark taggings, and query-click logs. Many of them can be viewed as 4-item tuples $\langle user, item, tag, time\rangle$ ($\langle u, d, w, t\rangle$), which means that at time t, user u annotates an item d using a word w. Take the search-engine query-click log for example. When a user clicks on a result, it can be considered as annotating the corresponding Web-resource URL with the query term(s) he issued. In this chapter, I analyze the delicious bookmark because of its wide availability to the public, while keeping the analysis model generic enough to be applicable in other social annotation domains as well.

I first investigate the bookmark frequency distribution. Not surprisingly, it follows a power-law-like distribution (Figure 5.1a); only a few URLs are bookmarked by (or known to) a large number of users, while the majority are bookmarked by only a few users. This is likely to be the result of a preferential



(a) URL

(b) Tag

Figure 5.1: Power-law distribution of URL/Tag usage frequency vs count.

attachment process [BA99] where existing popular URLs are bookmarked more over time. The word-frequency distribution also follows a power-law-like distribution, as in many text collections; one interesting thing to note is that the most frequently used words are *design*, *software*, and *tools* in the delicious dataset instead of words like *the*, *be*, and *to* in normal English text, indicating that social annotations differ significantly from the general text corpora assumed in common information-retrieval practice and require unconventional mining techniques.

Such a social annotation dataset provides invaluable information on users' perceptions of Web resources and clues to better categorize and rank these resources for retrieval. For example, Wu *et al.* [WZY06] and Zhou *et al.* [ZBZ08] both use the delicious bookmark data to improve the accuracy of information retrieval. For instance, when we collapse the data on the user and time dimensions, a URL d can be viewed as a document containing words that are the tags w annotated by different users u over time. When the annotation data is modeled in such a way, a document generative model, such as Latent Dirichlet Allocation

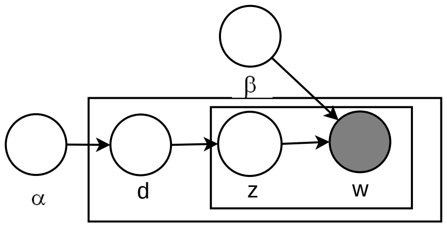


Figure 5.2: Plate notation of LDA.

(LDA) [BNJ03], can be applied to discover the hidden topics among the set of URLs being annotated.

The LDA model is based on the idea that the composition of words in a document are determined by an underlying generative process, in which, a document consists of a mixture of topics and a topic is represented as a probability distribution over words. The LDA process intends to recover the hidden topics and the parameters that govern this generative process. Figure 5.2 shows the plate notation of the LDA model, which illustrates the dependency between hidden and observable variables. α and β are the parameters of the uniform Dirichlet prior on the per-document topic distribution and the per-topic word distribution respectively; w_{ij} are the observable variables that indicate word w_j occurs in document d_i when given a collection of documents. When given a number of hidden topics, the LDA model infers the following two conditional probabilities:

- $p(z|d)$ - the topic distribution for document d
- $p(w|z)$ - the word distribution for topic z,

while the other two probabilities $p(d|z)$ and $p(z|w)$ referred to later in this chapter are then estimated by Bayes' Theorem.

Table 5.1 shows samples of 10 topics found by applying the LDA algorithm[2]

[2] A modified implementation of the LDA model by Steyvers and Griffiths [GS04] is used.

on the delicious bookmark data, with 250 hidden topics used. Each topic is listed together with the words having the highest $p(w|z)$ values, and the topic names are given manually after we examine the top associated words. Table 5.2 also shows the list of the top 5 URLs, those with the highest $p(d|z)$ values, in topic #103 (photography related).

In the above example, we see that social annotations, together with an appropriate analysis model, are useful and effective for Web-resource categorization. By the same token, it is also a good source of information on the word-usage behavior in the annotation process. Any usage pattern discovered from such a dataset will shed light on selecting advertising keywords automatically.

5.3 Desiderata of advertising keywords

An advertiser wishes to bid on keywords that will be most effective in bringing relevant traffic to his Website. In his search for these words, he may potentially ask the following questions:

- Do the words have specific semantic meanings covering topics in my domain of interest? Or are they too generic?
- What are the words that correspond to an emerging topic for which users are more likely to explore new pages? And what are the commonly used words that lead users to my competitors' pages instead of mine?
- Are the words sensitive to change in their meaning over time? For example, *holiday* may be associated more with Christmas gift shopping towards the end of the year but with vacation travel at other times. It will be useful to bid for if I advertise Christmas gifts, but what if I advertise travel packages

ID	topic	top tags
19	Copyright	copyright law legal creativecommons drm workflow rights freedom license cc
31	Viral marketing	marketing advertising ads viral pr ad affiliate branding commercials commercial
46	Autos	cars car auto beer alcohol automotive drinks silverlight party motorcycle
75	Freeware download	download downloads warez imported rapidshare cracks shareware crack descargas serial
103	Photography	photography photo camera photos digital photographer fotografia foto photoblog cameras
127	Sciences	science biology evolution nature chemistry death genetics interesting serials
154	Machine learning	ai simulation datamining applescript intelligence complexity bandslash algorithms bayesian machinelearning
184	Wikipedia	wiki wikis collaboration tiddlywiki mediawiki wikipedia information socialsoftware wikimedia
216	Computing security	security hacking password antivirus passwords spyware virus crack sysadmin seguridad
246	Adobe products	adobe vista air beta dreamweaver apollo lightroom labs fireworks macromedia

Table 5.1: Samples of topics found by LDA from delicious bookmark data.

Rank of $p(d\|z)$	URL
1	http://www.dpreview.com/
2	http://www.cambridgeincolour.com/tutorials.htm
3	http://www.digital-photography-school.com/blog/
4	http://www.flickr.com/
5	http://www.morguefile.com/archive/classroom.php

Table 5.2: Top 5 URLs with highest $p(d|z)$ values in Topic 103 (photography related).

for summer vacation? Will it still be effective?

With this in mind, the following sections present the analysis on the delicious bookmark in search of answers to the above questions.

5.3.1 Specific words

Since advertisers prefer reaching the target audience that is searching for their products or services, it is beneficial to select the correct set of advertising keywords that are *specific* in their domain of interest, rather than generic keywords that can be applied in many other contexts.

In information-retrieval common practice, a stop-word list is one technique used to filter out words that carry little or no conceptual meaning, such as *the*, *and*, and *or*. In addition, the inverse document frequency (*idf*), the logarithm of the ratio of the number of all documents to the number of documents containing a word, is used to demote words that are less distinguishable among documents in the $TF \times IDF$ document vector model when computing document-document or document-query similarity. However, social annotations differ substantially from the generally assumed document model, because the words are generally mnemonics carefully selected by users to characterize documents for easier retrieval. Moreover, the vocabulary is continually evolving, with new tags constantly being introduced [CM07]. It would become tedious to maintain a stop-word list to filter out words that have non-specific meanings. This drives us to explore new measures to characterize words based on the "specific" property.

As an output of the LDA model described in Section 5.2, a word can be expressed as a probability distribution over topics, $p(z|w)$, which indicates its degree of association with each hidden topic found. Such a distribution serves as an indicator of how likely a word is to be associated with many different concepts. By measuring the entropy of this distribution, as in Equation 5.1, where T is the number of topics, this metric captures how much a word is *spread out* across different concepts.

$$H(w) = -\sum_{i=1}^{T} p(z_i|w)log(p(z_i|w)) \tag{5.1}$$

Table 5.3 shows two samples of words that are found to be the least specific using the entropy measure and the traditional *idf* measure, which is inversely proportional to the number of documents annotated by the word, respectively. Despite the effectiveness of *idf* in demoting words with less distinguishing power in traditional document retrieval applications, this metric tends to select popular tags that still carry a specific meaning, such as *blog*, *music*, *web*, and *programming*, etc. The words selected by the entropy metrics are generic words like *temp*, *important*, *good*, and *misc* that can be applied in many different contexts.

Figure 5.3 shows the probability distribution over topics, $p(z|w)$, of 6 sample words, 3 from each list. Despite the words "web", "design", and "programming" having a low *idf* value, they all strongly associate with one or two topics only, which can be considered useful to advertisers in reaching the target audience in

Entropy	Tag	# URL tagged $\propto \frac{1}{idf}$	Tag
6.39	temp	178,837	imported
5.75	for	153,890	web
5.74	important	150,224	reference
5.74	and	129,919	software
5.62	good	128,781	design
5.47	imported	128,223	tools
5.24	tocheckout	118,870	blog
5.22	misc	104,520	programming
5.18	followup	100,773	toread
5.07	great	94,710	howto

Table 5.3: Sample of least specific words found by different metrics.

a search activity, but not the other three words *good*, *general*, and *other*.

5.3.2 Emerging vs. established

In addition to choosing specific words to reach the target audience, advertisers have an incentive to pick words that correspond to emerging topics within their domain and for which Internet users are willing to explore new pages.

There have been efforts in identifying trendy topics both in a search-engine query stream and from the Web content. For example, the increase in query frequency in a query stream, the increased usage of key phrases in an article collection such as the daily blog posts, or the daily fluctuation of these frequencies and any metrics that characterize the burstiness may all serve as indicators of emerging topics. While these techniques were proven useful in the litera-

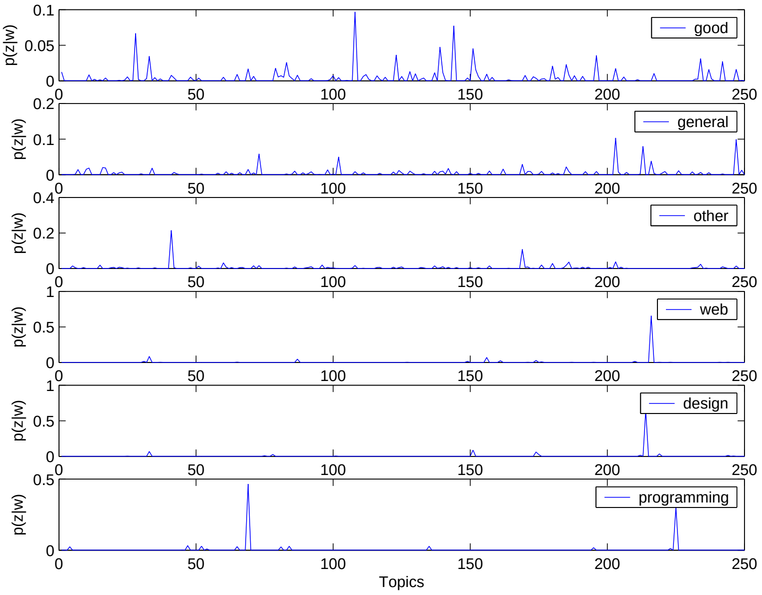


Figure 5.3: Distribution of $p(z|w)$ values of six word samples.

ture [GGL04, GHT04, AWB07], they were designed from the search engine's or Internet user's perspective and may not be directly applicable to online advertisers. For example, a sudden increase in the query volume of a newly released movie may not necessarily imply that people are exploring on this topic but just that they are visiting the official page for the trailer or for a few critics' Websites. From the online advertiser's perspective, a good set of keywords that correspond to emerging topics may need to satisfy at least the following two criteria:

- **Increase in usage**: Intuitively, an increase in either query frequency or tag usage in annotation implies that there is a large pool of Internet users interested in the associated topic, which will, in turn, guarantee a higher visibility of the advertisement placed.
- **Diversity**: The increase in usage may not be a good enough indicator alone. For example, an advertiser that sells e-books may prefer bidding on

kindle[3] than some other related terms such as *literature*, *palm*, *pdf*. Since Kindle is just released and emerging, people are keener to explore relevant pages on this topic than others; thus, the chance of an advertisement being clicked increases.

To capture the second criterion above, let us consider social annotations as a random process consisting of two discrete random variables, D and W. D represents a random variable whose sample space is the entire set of URLs, and W represents a random variable whose sample space is the entire set of tag vocabulary. Their probability mass functions are computed from the counts of page annotating and tag usage, respectively. For example, the value $P(W ='' www'')$ estimates the probability of a user using "www" to annotate any URL without any prior information given. When we consider the conditional entropy $H(D|W = w)$ of a given word w, it estimates how "diverse" documents being annotated by w are. This measure is indeed very useful to determine if a word corresponds to emerging topics as illustrated in the two examples below.

Example 6 Suppose a tag, w, is used to annotate two items, A and B, each with one annotation, at time t_1. Its conditional entropy, $H_{t_1}(D|W = w)$, at t_1 is measured as 1. Then at time t_2, two more users annotate item A with the same tag, w. The conditional entropy $H_{t_2}(D|W = w)$ now becomes 0.811. □

The above example shows that users tend to associate w with item A more than B, which means that, given a tag, w, the certainty of finding the best associated page increases over time. Indeed, such change is reflected as a decrease of the conditional entropy of w from 1 to 0.811, which deviates from the usual phenomenon that entropy tends to increases over time. Contrary to the

[3] Kindle is the e-book reading device introduced by Amazon in November 2007.

one above, the following example illustrates a scenario in which the conditional entropy increases.

Example 7 Suppose a tag, w, is used to annotate two items, A and B, once for each, at time t_1. The value $H_{t_1}(D|W = w)$ at t_1 is measured as 1. While at time t_2, two new items, C and D, appear and they are also annotated by the tag, w, once each. The conditional entropy now becomes 2. □

In this example, the tag, w, is used to annotate new pages rather than existing ones and is evenly distributed among them. It suggests that the topic associated with w is under development where there are no well-established authority pages corresponding to it, which makes users continue to explore different pages in search of a better one. In a probabilistic sense, such change implies that the uncertainty of finding good pages to match the given tag, w, has increased over time.

To further quantify this measure, the change in conditional entropy is divided by the logarithm of the change in the number of times a tag is used, which indicates how much entropy is changed per annotation made. Mathematically, this ratio is expressed as the following:

$$C(a, t_i) = |\langle u, d, w, t\rangle|, where \quad t < t_i, w = a \tag{5.2}$$

$$R(a, t_i, t_j) = \frac{H_{t_j}(D|W = a) - H_{t_i}(D|W = a)}{log_2(C(a, t_j) - C(a, t_i))}, \tag{5.3}$$

where $C(a, t_i)$ is the number of times a tag a had been used on or before t_i, and $R(a, t_i, t_j)$ is the entropy change rate of a tag, a, between t_i and t_j.

Figure 5.4 shows the conditional entropy change distribution between December 2007 and March 2008 of a set of popular tags (the top 40 words in each topic found by the LDA). It follows a typical bell-shaped distribution centered around

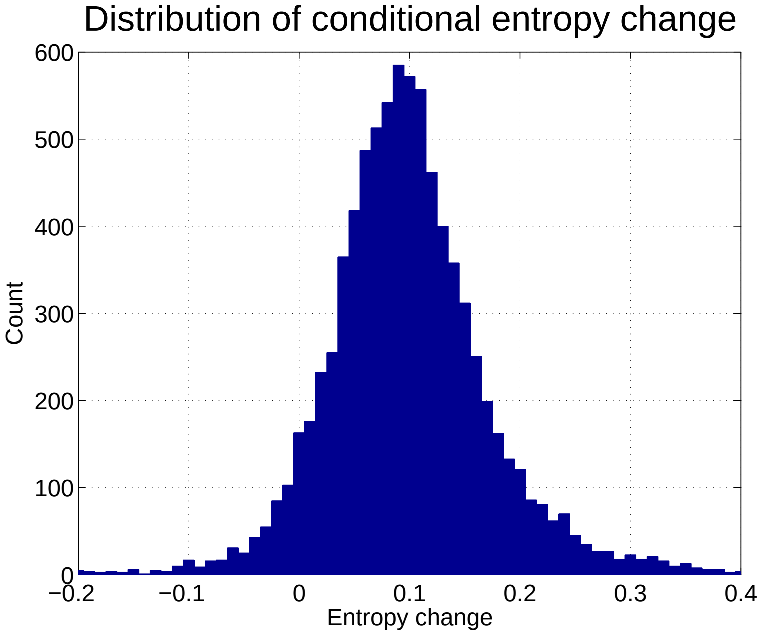


Figure 5.4: Distribution of entropy change of tags.

0.1[4]. Samples of tags in two extremes of the distribution are listed in Table 5.5 and 5.4, respectively.

Words listed in Table 5.4 all correspond to certain emerging topics within the investigation period, for example, "2008" (new year), "rails2.0" (Ruby on Rails 2.0 released on Dec 20, 2007), "kindle" (e-book device released by Amazon on Nov 19, 2007), and "obama" (Barack Obama and Hillary Clinton, Democratic presidential candidates). For words listed in Table 5.5, although there has been an increase of annotation usage, the conditional entropy decreases, which implies that annotations all go to the same set of pages. Examples include "arc" (annotating the Arc programming language official page `arclanguage.org`), "iplayer" (annotating the BBC iplayer programs page `www.bbc.co.uk/iplayer`), and "

[4] It implies that the conditional entropy of words increase by 0.1 over three months on average.

Tag	H_{t_1}	H_{t_2}	$C(w, t_1)$	$C(w, t_2)$
2008	8.03	10.67	3514	15038
bookmarksbar	9.11	10.95	605	2630
rails2.0	3.79	5.12	262	1266
kindle	3.86	5.14	610	1285
itouch	6.25	7.48	348	849
eeepc	5.07	6.22	968	4669
obama	5.41	6.54	1188	4134
jailbreak	5.45	6.44	907	2223
eee	5.40	6.39	701	2765
1.1.2	3.96	4.89	62	285

Table 5.4: Samples of tags with large conditional entropy change.

tomato" (annotating a guide to upgrade linksys router with tomato firmware `www.polarcloud.com/tomato`).

Besides using entropy change to identify good emerging keywords for advertisers, other metrics have also been studied, including the maximum daily annotation frequency change and percentage increase in tag usage, but their effectiveness is not as good. They, however, are also included these features in training classifiers, which will be discussed in Section 5.4.

5.3.3 Time sensitivity

As social annotations accumulate and new tags and documents are introduced into the collection, it is not surprising to see words that change in meaning over time. For example, when Christmas is approaching, the word *holiday*, which is normally associated with topics like vacation travel and airline booking, may be

Tag	H_{t_1}	H_{t_2}	$C(w, t_1)$	$C(w, t_2)$
openstandards	7.12	4.24	274	755
apology	4.27	2.25	87	480
arc	7.73	6.00	401	1097
Internet_tools	8.22	6.88	348	768
iplayer	3.78	2.49	188	486
deadpixel	3.67	2.44	68	364
r2d2	3.59	2.41	130	579
tomato	6.11	5.13	748	1335
screensharing	5.31	4.37	469	1012
dingbats	5.09	4.14	830	1459

Table 5.5: Samples of tags with negative conditional entropy change.

drifted towards a topic on Christmas holiday shopping. To ensure the efficacy of selected advertising keywords in a campaign and to identify new and useful words in a domain, it is crucial for advertisers to know which words are more susceptible to change in meaning over time, and which are more "stable."

I study two metrics in order to characterize the time-sensitive property of words, namely *Jaccard coefficient* and *KL-divergence*, in the following:

[**Jaccard coefficient**] I first attempt to quantify this property by examining how the set of *co-occurring words* of a word change. Co-occurring words are defined as the set of words that are used to annotate the same URL by the same user. Suppose A is the set of top K co-occurring words of w at time t_1, while B is the set at time t_2; the metric is defined as the Jaccard coefficient as follows:

$$\frac{|A \cap B|}{|A \cup B|}. \tag{5.4}$$

I assume that when a word changes its topic association, it will also be used together with a different set of words to annotate Webpages; thus, the above metric will give a lower score to such a *time-sensitive* word than a stable one.

[**KL-divergence**] As an output of the LDA model, a word can be represented by a probability distribution over topics, which indicates its degree of association with each of them. The change of a word's topic association can then be captured by comparing the difference of such distributions measured at different times. Consider the following example:

EXAMPLE 8 At time t_1, the probability distribution of a word w over 5 topics is $[0, 0, 0, 1, 0]$, respectively, while at time t_2, its probability distribution becomes $[0, 0.4, 0, 0.6, 0]$. Over time, w becomes more associated with topic #2 and less with topic #4. □

The above example shows that a word changes its association from one topic (#4) to two topics (#2 and #4) between t_1 and t_2. Such change are captured by measuring the difference between the two distributions. Suppose P_{t_1} denotes the probability distribution over topics of a word at time t_1, i.e. $P_{t_1} = \{p_{t_1}(z_i|w)|i = 1..T\}$, where T is the number of topics. The KL-divergence measure used to quantify the change is defined as follows:

$$\begin{aligned} D_{KL}(P_{t_1}||P_{t_2}) &= \sum_{i=1}^{T} p_{t_1}(z_i|w) log \frac{p_{t_1}(z_i|w)}{p_{t_2}(z_i|w)} \\ D &= \frac{D_{KL}(P_{t_1}||P_{t_2}) + D_{KL}(P_{t_2}||P_{t_1})}{2}. \end{aligned} \tag{5.5}$$

The computation of KL-divergence requires matching topics found at two different time points. For simplicity, topics are matched by ordering on the $p(z_i|w)$ values, under the assumption that topic association will not change drastically over time. Moreover, since there is randomness involved in the LDA process,

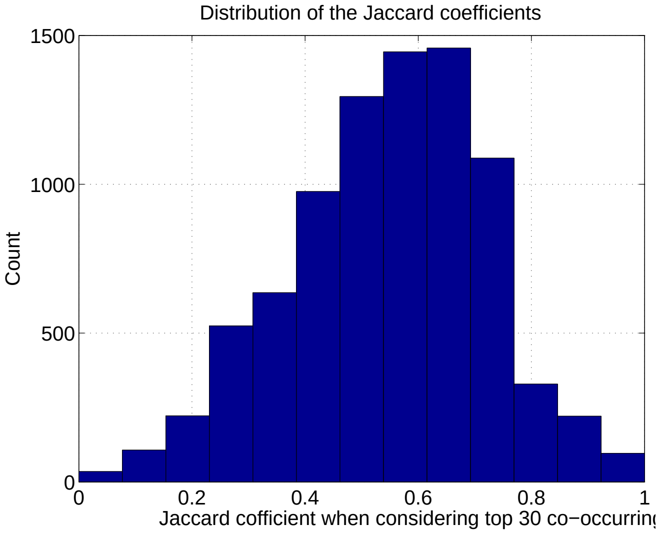


Figure 5.5: Distribution Jaccard coefficients when considering top 30 co-occurring words.

multiple runs of LDA are performed and the median of the D values is used to characterize how likely a word is to change its topic association over time.

Figures 5.5 and 5.6 show the distribution of the Jaccard coefficient and KL-divergence measures. It is observed that the majority of the words may not be sensitive to change in meaning over time, as the percentage of words with low Jaccard coefficient and high KL-divergence is comparatively small.

I present an example from the set of tags that are considered to be sensitive to change in meanings based on the KL-divergence measure. Figure 5.7 shows the probability distribution over 100 topics of the word *programmers* in October 2005 and January 2006, respectively. (The top three graphs are different runs of LDA in October, while the bottom three graphs are in January). It was originally associated with one topic (programming related), but after three months, it became associated with an additional topic (career and job hunting related). This

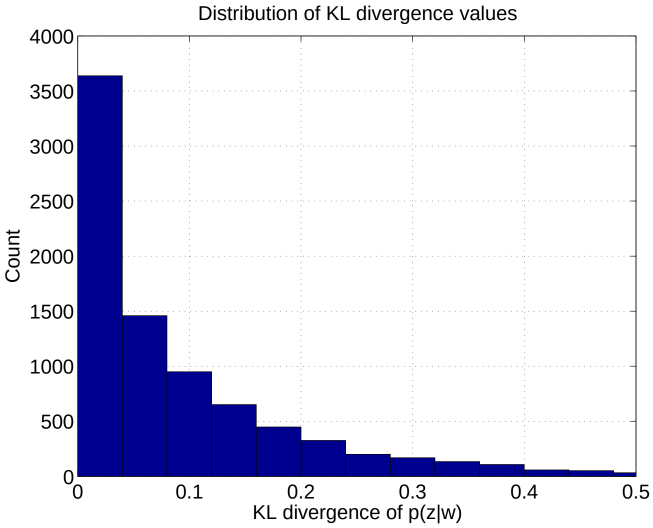


Figure 5.6: Distribution of KL-divergence of popular tags.

can be explained by the hiring season of college graduates during that period, where users tend to associate *programmers* with the concept of job hunting. In this example, we see that it is crucial for online advertisers to identify words with such time-sensitive properties either to avoid (for those who advertise programming tools) or emphasize (for those who advertise for job hunting) in the bidding process, depending on their domain of interest.

5.4 Experiments

In this section, I describe how to build a word classifier that answers these questions: (1) Does a word have a specific meaning? (2) Does a word represent a hot topic? (3) Are the concepts associated with a word stable over time? First, I describe the process of obtaining the annotation data and the set of features extracted to represent a word. Then, I describe a user experiment that captures the ground truth of users' judgments on words. Lastly, I present the details of

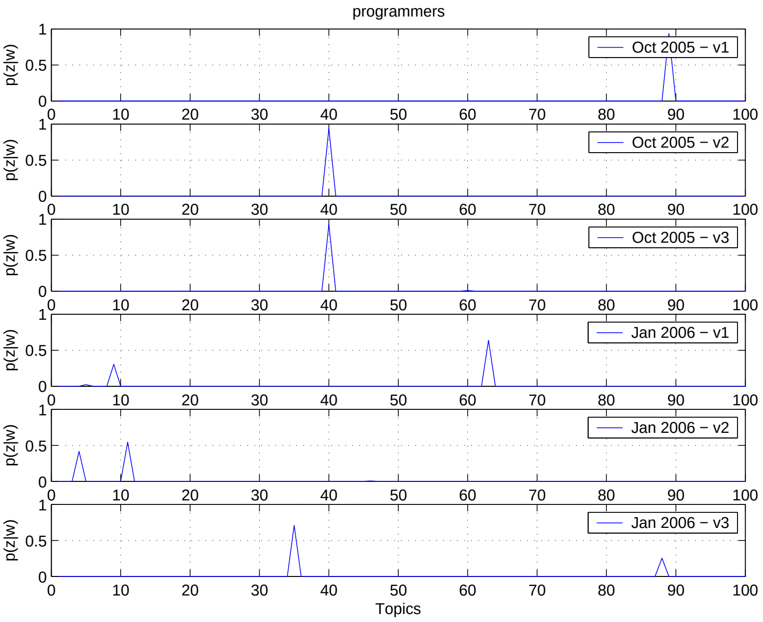


Figure 5.7: The topic association probability distribution at different time periods.

building such a classifier and report the results.

5.4.1 Data Preparation

Using a set of popular URLs[5] in February 2008, I started crawling the *delicious* Website, from which a set of users that have annotated these URLs is collected. By iteratively following the set of URLs tagged by users and users who annotated the URLs, a dataset consisting of 322,657,413 annotations made by 600,000+ users on 36,435,340 URLs, using 4,560,695 distinct tags is obtained. The URLs are normalized by some basic rules like removing the trailing "/" and "/index.html," and the tags are all lower-cased. After filtering out the unpopular URLs and tags that had been annotated or used fewer than 20 times in the entire collection, it resulted in a dataset consisting of 537,114 URLs and 346,555 tags to be used in the experiment. The growth of the *delicious* bookmark data from 2004 to 2007 is shown in Figure 5.8. Based on the sampled data, *delicious* was growing very fast in the years 2004 and 2005 and has slowed down to a linear growth since 2006. In addition, the daily number of bookmarks also follows a regular weekly pattern, with substantially more bookmarks on weekdays than weekends.

5.4.2 Feature extraction

As discussed in Section 5.3, there are several key features that characterize the three properties well. The following additional features are also included in order for a classifier to pick up all necessary signals to determine a word's properties.

1. *NUM_PAGE*: The number of *unique* URLs annotated within a period. This

[5] Those URLs listed on the *delicious* Website as popular bookmarks.

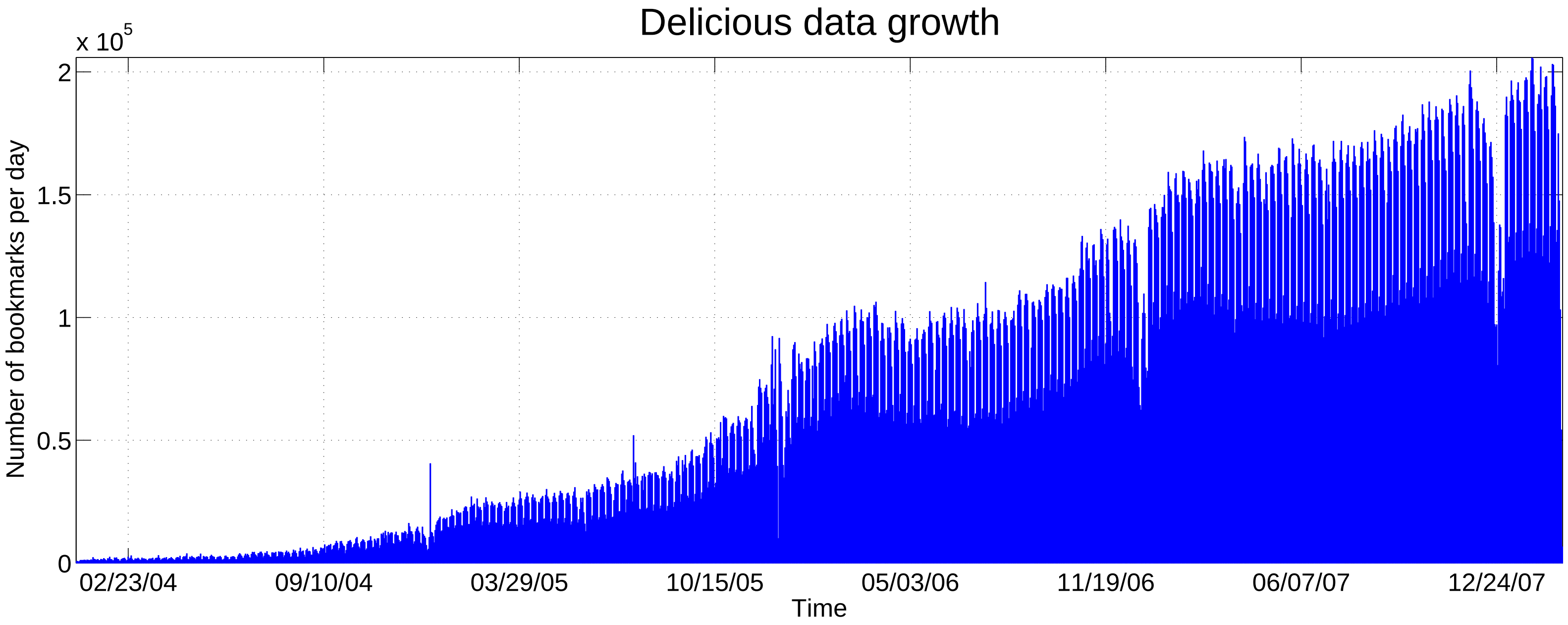


Figure 5.8: Distribution of bookmark data over time.

feature follows the intuition that a word is likely to be non-specific if it is used to annotate many different items, which is similar to the inverse document frequency concept.

2. *PZW_ENT*: The entropy measure of the $p(z|w)$ distribution of a word, as discussed in Section 5.3.1.

3. *ENT_CHG*: The change in conditional entropy $H(D|W = w)$, discussed in Section 5.3.2.

4. *ENT_CHG_RATIO*: The relative change in conditional entropy, which attempts to factor in the base conditional entropy as well.

5. *BURST*: The ratio of maximum daily change in annotation counts to the total annotation counts within a period. It tries to capture the scenario where a surge in word usage may correspond to emerging events.

6. *KL_DIV*: The KL-divergence of two $p(z|w)$ distributions measured at different times, as discussed in Section 5.3.3.

7. *TOPIC_CHG*: The change in the number of topics having $p(z|w) > 0.1$ across a period of time. If a word is sensitive to changing topic association, this count may also change when it is associated with more or fewer topics over time.

8. *COOC_JACCARD*: The amount of overlap between the top 30 co-occurring words at different times as discussed in Section 5.3.3.

9. *ASSOTAG_JACCARD*: The amount of overlap between *associated tags* at different times. Associated tags refer to the set of top 30 tags with highest $p(w|z)$ values in the topics z with which a word is associated (i.e. $p(z|w)$

> 0.1). This is similar to co-occurring tags above, but applied in the LDA domain.

5.4.3 Web-user evaluation experiment

To obtain the ground truth and a training dataset to build the word classifier, I design an experiment that asks users to judge words based on three questions that correspond to the properties in Section 5.3 (1) Does it have a specific meaning? (2) Does it represent a hot topic? (3) Is the concept associated with it stable over time?

5.4.3.1 Pairwise comparisons

Because it may sometimes be difficult to judge the properties given one single word alone, the experiment, instead, presents pairs of words and ask users to compare them based on the three properties in question, using the following procedure:

1. 420 words are sampled to be evaluated. Words are sampled from those with extreme values, both in the high and low range, among the features that are considered previously (to ensure that we cover the entire spectrum of words in the three properties under investigation), and some are sampled randomly.

2. The users are shown with instructions and examples describing the three properties that they are going to judge.

3. The users are then presented 5 pairs of words at a time and asked to finish a questionnaire that has four options, as shown in Figure 5.9. Take the specific question about specificity, for example. Choosing option A for the

Word A	Word B	Question	A	B	C	D
superbowl	**advanced**	More specific?	⊗	○	○	○
		Hot topic?	⊗	○	○	○
		Time-sensitive?	○	○	⊗	○

Figure 5.9: A sample pair-wise comparison question and its answers.

“More specific?” question indicates that Word A, “superbowl,” has a more specific meaning than Word B,“advanced,” and option B is the opposite. Option C indicates that they are similar, while option D allows users to skip this question when they do not understand either one of the words.

4. Since a complete pair-wise comparison is expensive, 1 word is compared with 10 other randomly selected words in the pool. For each pair, 3 distinct users are selected as judges. Answers with option D are discarded, which covers 12% of the total judgments.

To expedite the process of gathering user judgments, the questionnaires is submitted to the Amazon Mechanical Turk platform [Mtu], which is a marketplace for outsourcing tiny tasks, typically repetitive and labor-intensive work that is done better by humans than computers, for example, object recognition from pictures. A reward of USD $0.02 is given for every five questions and gathered 1,263 pairs of judgments in roughly 66 hours. Figure 5.10 shows the time distribution to complete an assignment. The average time used to finish five questions is 77 seconds. In order to ensure the quality of labeling, answers that are completed in fewer than or exactly 30 seconds are rejected.

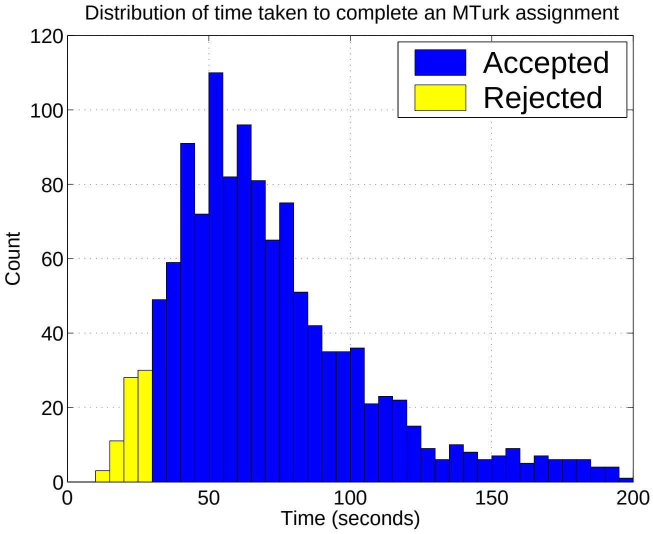


Figure 5.10: Distribution of the time spent on each MTurk assignment.

5.4.3.2 Training of classifiers

To combine all pair-wise comparisons into a single ranking of words on each of the three properties, I attempted the following two methods:

- **Topological sort** : Options A and B in the questionnaires generate a partial ordering, $w1 \prec w2$, between the two words under comparison. Suppose we gather three pairwise comparisons, like $w1 \prec w2$, $w2 \prec w3$, and $w2 \prec w4$; the topological sort will generate a global ranking, $w1 \prec w2 \prec w3 \prec w4$, from which we can order the words based on each of the properties investigated. To ensure the quality of the partial ordering judgments, only those with at least two distinct users in agreement on option A or B are considered.

- **Scoring** : A score of 1 is assigned for option A, 0 for option B, and 0.5 for option C. By taking the average score of words for ten pair-wise com-

Least specific	Most emerging	least stable	most stable
great	jailbreak	fav	websearch
for	socialnetworking	kde4	official
totry	wiki	eeepc	subway
howto	beef	rasterbator	comics
the	airfare	colinux	software
fav	ipodtouch	lpi	animation
todo	timemachine	jsonp	baseball

Table 5.6: Samples of words found using the scoring based method.

parisons, we generate a global ranking on each of the three properties. The distribution of the average scores is shown in Figure 5.11. While the majority of words have scores around 0.5, we select words in the two extremes, those having scores > 0.6 and < 0.4, as the positive and negative training samples, respectively.

While topological sort and scoring both produce similar ranking, the scoring method is used for its simplicity and stability[6]. Table 5.6 shows samples of tags with extreme values on each of the properties based on the scoring approach. Using this method, a training dataset is compiled for each property, where the class distribution is listed in Table 5.7.

For each of the training datasets, 7 different classifiers in Weka [Wek] are used, namely, SVM, Simple Logistic regression, C4.5 classification tree, Random forest, K-nearest neighbors, Back-propagation neural network, and Naive Bayes classifiers. To further improve the performance, we combine the best 5 classifiers

[6] Topological sort may generate many possible solutions given the same set of partial orderings.

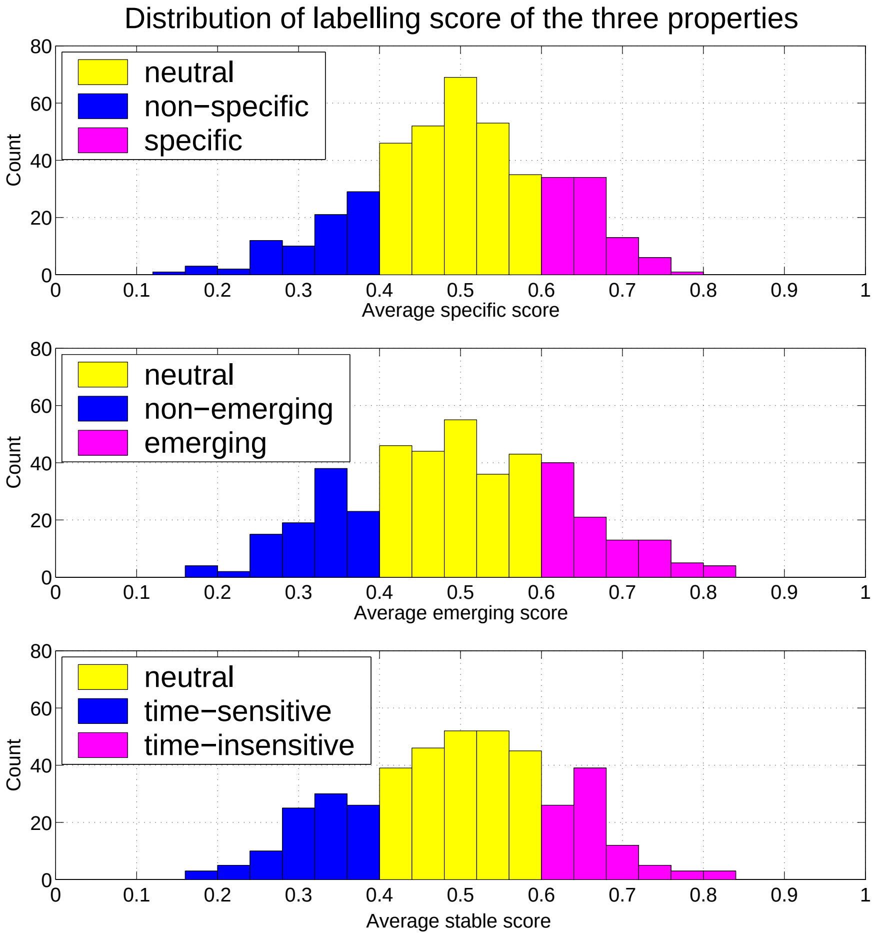


Figure 5.11: Distribution of the score obtained in the labeling of the three properties.

Classes	Specific	Emerging	Stable
Positive	91	103	95
Negative	78	99	88

Table 5.7: Distribution of training data class label after thresholding on scoring method.

Classifier	Specific	Emerging	Stable
SVM	65.7%	70.3%	63.9%
Logistic regression	66.3%	69.8%	60.7%
C4.5	64.5%	**73.3%**	59.0%
Random forest	65.1%	**73.3%**	57.4%
K-nn (k=5)	60.1%	67.3%	**64.5%**
Multilayer perceptron	60.3%	63.9%	60.1%
Naive Bayes	**67.4%**	59.9%	63.4%
Best 5 combined	66.3%	73.8%	63.4%

Table 5.8: Performance of individual and combined classifiers in 10-fold cross-validation.

using the majority-vote method. The performance on a 10-fold cross-validation is listed in Table 5.8. The accuracy is not as good as it is expected to be. This is partially due to the labeling quality in the training dataset where we have little control over the "real" effort spent by MTurk users. Nonetheless, it serves as a good starting point for incorporating additional information sources in effective advertising keywords selection. Additional features may be added to improve the overall classification accuracy in the future.

5.4.3.3 Correlation analysis

Based on the same training dataset obtained in the user experiment, I apply the Logistic regression model to identify features that correlate most with the three advertising keyword properties among the ones described in Section 5.4.2. Table 5.9 shows the top four features that have the largest absolute regression coefficient in the Logistic regression model. It shows that *specific* correlates most

Property	Features	Regression coef.
Specific	PZW_ENT	-0.24
Emerging	COOC_JACCARD	-0.96
	PZW_ENT	-0.38
	KL_DIV	-0.23
	ENT_CHG	0.23
Stable	ENT_CHG	-0.35
	ASSOTAG_JACCARD	-0.29
	TOPIC_CHG	0.26
	KL_DIV	-0.24

Table 5.9: Top four correlated features in each property.

with *PZW_ENT*, but not with anything else. The correlated features of *emerging* and *time-sensitive* are tightly coupled, which indicates that they are somewhat interchangeable. For example, an emerging keyword can also be considered as time-sensitive to change in meanings because it might be associated with different subtopics as the topic develops. Nonetheless, the result still indicates important features to consider when selecting effective advertising keywords.

5.5 Related Work

With the emergence of online sponsored search advertising, there is much ongoing research in different research communities, such as economics, social networks, and information retrieval.

Based on a game-theoretic approach, mainly driven by the economics community, the generalized second-price-auction method [EOS07] proposes to maximize the income of a search engine by application to the keyword bidding process of sponsored search activities. Recently, Auerbach *et al.* [AGS08] have also studied the bidding behavior of advertisers in maximizing their return on investment. Viral marketing is also a closely related area where researchers focus on optimizing the placement of advertisements in a network in order to maximize the overall impact [DR01, RD02].

In the information-retrieval community, there has been research on selecting advertisement keywords on Webpages to match with advertisements, as in the contextual advertisement model [YGC06, AFJ07, WB08], while research on the sponsored search model mainly focuses on understanding the commercial intention of queries [DZN06] and matching relevant advertisement snippets with the queries [RDR07]. I believe my work complements existing literature by exploring additional information sources for more effective advertising keywords selection.

Social annotation begins to play an important role in the Internet search activities since it involves more user participation than ever. Recently, Wu *et al.* [WZY06] and Zhou *et al.* [ZBZ08] have studied how to leverage the delicious bookmark data, using a document generative model, to improve the relevance of information retrieval on the Web. In addition, Millen and Feinberg [MF06] have also applied it to assist information sharing within an organization. Chi and Mytkowicz [CM07] have studied the bookmark data in an information theoretic approach to show that its usability as an information-retrieval tool alone decreases over time. Nevertheless, social annotation still remains a powerful tool to organize Web resources.

One aspect of the proposed keyword selection method focuses on detecting emerging topics. There has been research on detecting *trendy* topics on the Web, using variants of phrases occurring frequency on blog and news content. [GGL04, GHT04]. Recently, Adar *et al.* [AWB07] attempt to correlate query streams with blog and news content to understand the behavioral properties of Web searches. I believe the study on social annotations not only benefits advertisers in choosing better keywords but also improves on identifying trendy topics on the Web.

5.6 Summary

In both contextual advertising and sponsored search activities, considerable research has been focusing on finding the best matching advertisement to a Webpage or a search query respectively. Consequently, search engines maximize their revenue and improve user experience by providing fewer annoying advertisements. However, there is not much emphasis on the advertiser's perspective in determining better advertising keywords and to incorporate additional information sources, such as user-generated content.

In this chapter, I studied social annotations, collected from users' interactions with an online content aggregator, as an extra information source for analyzing word-usage behavior on the Web. In addition to its application in Web-resource categorization and information retrieval. I aimed to utilize this information in the domain of effective advertising keyword selection. I discussed three criteria, namely, specificity, emergence, and time-sensitivity, that observed from the tag usage of an online social bookmark Website, for selecting effective advertising keywords. I have also applied features extracted from social annotation data to characterize them. I conducted an online experiment to solicit user's judgment on words and build a classifier for identifying keywords based on three properties with fairly good accuracy. A correlation analysis also revealed the suitable features to characterize each of the criteria.

CHAPTER 6

Conclusion

With the growing popularity of "Web 2.0" services, it has become easy and convenient for less technically savvy users to publish content on the Web, leading to the explosion of user-generated online content. However, such an increase in quantity does not guarantee an improvement in the quality. Oftentimes, users find themselves overwhelmed by the magnitude of new information being generated every day.

Researchers and engineers have been building online content aggregators that help users better manage their subscribed RSS feeds and tap into Web 2.0 information easily. Such aggregators, when supporting a large number of users with diverse interests and subscriptions, face many challenges and opportunities such as: delivering updated content, providing good personalization efficiently and accurately, and datamining user generated content for improving the system.

In this dissertation, through an analysis of existing online content aggregators, I studied the challenges and opportunities in delivering fresh and personalized content to users and how to make use of the data collected from users' interactions with the systems. More specifically, I addressed the following technical challenges in this dissertation:

- **Monitoring RSS feeds** - I studied the content generation and user-access patterns of RSS feeds and showed that these patterns, when investigated at a finer time scale, are different from what was previously assumed in the literature. The observed patterns are often fluctuating (within one day) and repeating (everyday), which are best modeled by the periodic inhomogeneous Poisson process as proposed in this dissertation.

After developing the new model based on this observation, I proposed an optimal RSS-feed monitoring policy that significantly improves the "freshness" of the content delivered to the users in resource-constrained settings. The proposed policy utilizes better resource allocation (i.e. allocating more retrievals to sources of higher importance, such as those with more subscribers or update more frequently), and better retrieval scheduling (i.e. scheduling retrievals closer to source updates to shorten the delay). I have demonstrated its effectiveness and properties through experiments with a real-life dataset, which is collected from 10k RSS feeds (data posting patterns) and a few volunteers (user access patterns), and also investigated the potential improvement brought by a proposed change in the RSS protocol.

- **Ranking of articles** - In order for online-content-aggregator users to keep up with the growing number of new articles being generated every day in her subscription list, oftentimes, an aggregator provides a ranked list of articles based on the user's interest on her front page. Generating this ranked list requires the system to learn the user's interests from her previous online activities because users are often reluctant to express their interests or provide any feedback explicitly as shown in the literature.

  I, together with my collaborators, introduced a learning process that uses a probabilistic framework to model the user reading and clicking behavior of articles in a ranked list. Based on this framework, we introduced an *Exploitation* (weighting topics that are known to be a user's interest higher) and *Exploration* (weighting topics that are less shown to a user higher) strategy to rank articles among different topics in order to learn a user's profile quickly through an iterative process.

While the results are not conclusive due to the potential bias and small size of our experiments, through simulations and a small-scale pilot user study, we showed that the proposed *E&E* strategy has the potential to improve the likelihood of a user clicking articles in the ranked list and to learn the user profile more quickly and accurately when compared to a *Exploitation-only* greedy approach.

- **Efficient personal recommendations** - As an additional functionality for online content aggregators to improve user experience, it may analyze the content within all subscribed RSS feeds of a user and recommends popular key phrases and Web links, that are being mentioned frequently in the subscribed sources, for her further investigation. In Chapter 4, I studied the problem of making such personalized recommendations based on users' RSS feed subscriptions and addressed on the scalability challenges faced by these systems in supporting a large number of users with diverse individual interests.

  I started by modeling the personal recommendation computation as matrix operations and observed that users often share similar interests, hence, subscription lists. I applied the *Non-negative Matrix Factorization* to cluster users into different interest groups based on their subscription similarity; thus, the computation can be separated into two stages, precomputation of results for user groups and combination of precomputed results for individual users. The process is further speeded up by applying a top-k computation algorithm, *Threshold Algorithm.*

I showed that it is possible to compute personalized recommendations for users at a much lower cost than by existing methods and discussed the adjustment of parameters to further improve the efficiency. I also demonstrated that personalized recommendations generate results that are very different from non-personalized recommendations and among users.

- **Social annotation analysis** - When such an aggregator grows over time, it collects a huge amount of user interaction data within the system, which is often in the form of social annotations. It involves tremendous human efforts in matching the best descriptive keywords with the information sources on the Web. Such data, when utilized appropriately, can greatly improve the Web information retrieval process.

  As an example of applying this rich body of data to help users, I studied the evolution of the online tagging data to select effective advertising keywords. I first applied the *Latent Dirichlet Allocation* to discover hidden topics among Web resources and their associated keyword membership. By taking into account other features, such as entropy change of tags and diversity of tag membership, in addition to the topic membership discovered through Latent Dirichlet Allocation, I investigated three properties of keyword usage in tagging, namely: specificity, emergence, and time-sensitivity.

  These properties match closely with the criteria of choosing effective keywords in an advertising campaign. Using a dataset collected from an online social bookmarking service, *delicious*, I explored on combing existing classifier methods to automatically detect keywords with such properties from the evolving tagging data and illustrated the best corresponding features in identifying each property mentioned.

Through the investigations presented in this dissertation, I aim to solve the engineering challenges faced by building a personalized online content aggregator and explore the opportunities of mining the collected user interaction data. My hope is that, the methods and results presented in this dissertation not only help researchers build better online content aggregators but also provide insight for upcoming researches on information retrieval of Web 2.0 user-generated content.

The primary focus of this dissertation is the investigation of technical challenges, so some of the experiments that are presented in this dissertation do not provide a generally applicable conclusion due to the potential bias in the sample selection and the small size of the participants. It will be an interesting future work to extend these experiments and validate the effectiveness of some of the approaches proposed in this dissertation in more general settings and explore further issues discovered through this process.

References

[AA05] Eytan Adar and Lada A. Adamic. "Tracking Information Epidemics in Blogspace." In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2005.

[ABD06a] Eugene Agichtein, Eric Brill, and Susan Dumais. "Improving Web Search Ranking by Incorporating User Behavior Information." In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[ABD06b] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. "Learning User Interaction Models for Predicting Web Search Result Preferences." In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[AF00] Mehmet Altmel and J. Michael Franklin. "Efficient Filtering of XML Documents for Selective Dissemination of Information." In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2000.

[AFJ07] Broder Andrei, Marcus Fontoura, Vanja Josifovski, and Lance Riedel. "A Semantic Apporach to Contextual Advertising." In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.

[AGS08] Jason Auerbach, Joel Galenson, and Mukund Sundararajan. "An Empirical Analysis of Return on Investment Maximization in Spnsored Search Auctions." In *Proceedings of the Second International Workshop on Data Mining and Audience Intelligence for Advertising (ADKDD)*, 2008.

[AWB07] Eytan Adar, Daniel S. Weld, Brian N. Bershad, and Steven D. Gribble. "Why We Search: Visualizing and Predicting User Behavior." In *Proceedings of the International World-Wide Web Conference (WWW)*, May 2007.

[BA99] Albert-Laszlo Barabasi and Reka Albert. "Emergence of Scaling in Random Networks." *Science*, **286**:509–512, 1999.

[BC00] B.E. Brewington and G. Cybenko. "How Dynamic is the Web." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2000.

[Bloa] "Blogger." `http://www.blogger.com`.

[Blob] "Bloglines." `http://www.bloglines.com`.

[Bloc] "Blogpulse." `http://www.blogpulse.com`.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." *Journal of Machine Learning Research*, **3**:993–1022, 2003.

[BNW03] Andrei Z. Broder, Marc Najork, and Janet L. Wiener. "Efficient URL Caching for World Wide Web Crawling." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2003.

[Bro02] Andrei Z. Broder. "A Taxonomy of Web Search." *SIGIR Forum*, **36**(2), 2002.

[CDT00] Jianjun Chen, David J. DeWitt, Feng Tian, and Yuan Wang. "NiagaraCQ: A Scalable Continuous Query System for Internet Databases." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2000.

[CG00] Junghoo Cho and Hector Garcia-Molina. "Synchronizing a Database to Improve Freshness." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2000.

[CG02] Junghoo Cho and Hector Garcia-Molina. "Parallel Crawlers." In *Proceedings of the International World-Wide Web Conference (WWW)*, Honolulu, Hawaii, May 2002.

[CG03a] Junghoo Cho and Hector Garcia-Molina. "Effective Page Refresh Policies for Web Crawlers." *ACM Transactions on Database Systems*, **28**(4), 2003.

[CG03b] Junghoo Cho and Hector Garcia-Molina. "Estimating Frequency of Change." *ACM Transactions on Internet Technology*, **3**(3), August 2003.

[CLW98] Edward G. Coffman, Jr., Zhen Liu, and Richard R. Weber. "Optimal Robot Scheduling for Web Search Engines." *Journal of Scheduling*, **1**(1), 1998.

[CM07] Ed H. Chi and Todd Mytkowicz. "Understanding Navigability of Social Tagging Systems." In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, February 2007.

[CN02] Junghoo Cho and Alexandros Ntoulas. "Effective Change Detection Using Sampling." In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2002.

[CPC06] Jennifer Chu-Carroll, John Prager, Krzysztof Czuba, David Ferrucci, and Pablo Duboue. "Semantic Search via XML Fragments: A High-Precision Approach to IR." In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2006.

[DDG07] Abhinandan Das, Mayur Datar, and Ashutosh Garg. "Google News Personalization: Scalable Online Collaborative Filtering." In *Proceedings of the International World-Wide Web Conference (WWW)*, May 2007.

[Del] "Delicious bookmark." `http://delicious.com/`.

[DKP01] Pavan Deolasee, Amol Katkar, Ankur Panchbudhe, Krithi Ramamritham, and Prashant Shenoy. "Adaptive Push-Pull: Disseminating Dynamic Web Data." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2001.

[DLP06] Chris Ding, Tao Li, Wei Peng, and Haesun Park. "Orthogonal Nonnegative Matrix Tri-factorizations for Clustering." In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2006.

[DMO] "Dmoz Open Directory Project." `http://www.dmoz.org`.

[DPS04] Josep Domenech, Ana Pont, Julio Sahuquillo, and Jose A. Gil. "An Experimental Framework for Testing Web Prefetching Techniques." In *the 30th EUROMICRO Conference*, 2004.

[DR01] Pedro Domingos and Matt Richardson. "Mining the Network Value of Customers." In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2001.

[DZN06] Honghua Dai, Lingzhi Zhao, Zaiqing Nie, Ji-Rong Wen, Lee Wang, and Ying Li. "Detecting Online Commercial Intention (OCI)." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2006.

[Eft00] E. N. Efthimiadis. "Interactive Query Expansion: A User-based Evaluation in a Relevance Feedback Environment." *Journal of the American Society for Information Science*, **51**(11), 2000.

[EGS98] Abdulmotaleb El-Saddik, Carsten Griwodz, and Ralf Steinmetz. "Exploiting User Behavoiour in Prefetching WWW Documents." In *IDMS*, pp. 302–311, 1998.

[EMT00] Jenny Edwards, Kevin McCurley, and John Tomlin. "An Adaptive Model for Optimizing Performance of an Incremental Web Crawler." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2000.

[EOS07] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. "Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords." *American Economic Review*, **97**(1):242–259, March 2007.

[FJL01] Francoise Fabret, Arno Jacobsen, Francois Llirbat, Joao Pereira, and Ken Ross. "Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2001.

[FLN01] Ronald Fagin, Amnon Lotem, and Moni Naor. "Optimal Aggregation Algorithms for Middleware." In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 2001.

[GBP05] Joshua Grossnickle, Todd Board, Brian Pickens, and Mike Bellmont. "RSS–Crossing into the Mainstream." Technical report, Yahoo, October 2005.

[GCS95] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis.* Chapman & Hall, New York, 1995.

[GE01] Avigdor Gal and Jonathan Eckstein. "Managing Periodically Updated Data in Relational Databases: A Stochastic Modeling Approach." *Journal of the ACM*, **48**(6):1141–1183, 2001.

[GGL04] Daniel Gruhl, R. Guha, David Liben-Nowell, and Andrew Tomkins. "Information Diffusion Through Blogspace." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2004.

[GHT04] Natalie S. Glance, Matthew Hurst, and Takashi Tomokiyo. "BlogPulse: Automated Trend Discovery for Weblogs." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2004.

[GJ74] J. C. Gittins and D. M. Jones. “A Dynamic Allocation Index for the Sequential Design of Experiments.” In J. Gani *et al*, editor, *Progress in Statistics*, volume I, pp. 241–66. North-Holland, Amsterdam-London, 1974.

[Gooa] “Google Adwords.” `http://adwords.google.com/`.

[Goob] “Google Alerts.” `http://www.google.com/alerts`.

[Gooc] “Google Reader.” `http://reader.google.com`.

[good] “Google Sitemaps Protocol Beta.” `https://www.google.com/webmasters/sitemaps/docs/en/about.html`.

[GS04] Thomas L. Griffiths and Mark Steyvers. “Finding Scientific Topics.” *Proceedings of the National Academy of Sciences*, **101**(suppl. 1):5228–5235, 2004. Toolbox `http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm`.

[Hij99] Yoshinori Hijikata. “Estimating a User’s Degree of Interest in a Page during Web Browsing.” In *Proceedings of Systems, Man and Cybernetics Conference*, pp. 105–110, 1999.

[HKT04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. “Evaluating Collaborative Filtering Recommender Systems.” *ACM Transactions on Information Systems*, **22**(1):5–53, 2004.

[Hos05] Kartik Hosanagar. “A Utility Theoretic Approach to Determining Optimal Wait Time in Distributed Information Retrieval.” In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[Hoy04] Patrik Hoyer. “Non-negative Matrix Factorization with Sparseness Constraints.” *Journal of Machine Learning Research*, **5**:1457–1469, 2004.

[Jam05] Anthony Jameson. “User Modeling Meets Usability Goals.” In *Proceedings of the 10th International Conference on User Modeling*, 2005.

[JKF07] Akshay Java, Pranam Kolari, Tim Finin, Anupam Joshi, and Tim Oates. “Feeds That Matters: A Study of Bloglines Subscriptions.” In *Proceedings of ICWSM*, Boulder, Colorado, USA, March 2007.

[Joa02] Thorsten Joachims. "Optimizing Search Engines using Clickthrough Data." In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2002.

[JZM05] Xin Jin, Yanzan Zhou, and Bamshad Mobasher. "Task-Oriented Web User Modeling for Recommendation." In *Proceedings of the 10th International Conference on User Modeling*, 2005.

[KB04] Diane Kelly and Nicolas J. Belkin. "Display Time as Implicit Feedback: Understanding Task Effects." In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2004.

[KDF05] Diane Kelly, Vijay Deepak Dollu, and Xin Fu. "The Loquacious User: A Document-Independent Source of Terms for Query Expansion." In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[KI05] Georgia Koutrika and Yannis Ioannidis. "Constrained Optimalities in Query Personalization." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2005.

[KQ04] David R. Karger and Dennis Quan. "What Would It Mean to Blog on the Semantic Web." In *The Internaional Semantic Web Conference*, 2004.

[KWF06] Sailesh Krishnamurthy, Chung Wu, and Michael Franklin. "On-the-fly Sharing for Streamed Aggregation." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2006.

[LC05] Jie Lu and Jamie Callan. "Federated Search of Text-Based Digital Libraraies in Hierarchical Peer-to-Peer Networks." In *Proceedings of the European Conference on Information Retrieval (ECIR)*, 2005.

[LCI06] Chengkai Li, Kevin Chen-Chuan Chang, and Ihab Ilyas. "Supporting Ad-hoc Ranking Aggregates." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2006.

[Lin07] Chih-Jen Lin. "Projected Gradient Methods for Non-negative Matrix Factorization." *Neural Computation*, **19**(10):2756–2779, October 2007.

[LLC05] Uichin Lee, Zhenyu Liu, and Junghoo Cho. "Automatic Identification of User Goals in Web Search." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2005.

[LM03] Ronny Lempel and Shlomo Moran. "Preditive Caching and Prefetching of Query Results in Search Engines." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2003.

[LPT99] Ling Liu, Calton Pu, and Wei Tang. "Continual Queries for Internet Scale Event-Driven Information Delivery." *IEEE TKDE*, **11**:610–628, 1999.

[LWL07] Chengkai Li, Min Wang, Lipyeow Lim, Haixun Wang, and Kevin Chen-Chuan Chang. "Supporting Ranking and Clustering as Generalized Order-By and Group-By." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2007.

[Mar96] Evangelos P. Markatos. "Main Memory Caching of Web Documents." In *Proceedings of the International World-Wide Web Conference (WWW)*, 1996.

[mas06] Duane Merrill. "Mashups: The new breed of Web app.", 2006. `http://www-128.ibm.com/developerworks/library/x-mashups.html`.

[MCS00] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. "Automatic Personalization Based on Web Usage Mining." *Communications of the ACM*, **43**(8), 2000.

[MDL02] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. "Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization." *Data Mining and Knowledge Discovery*, **6**(1), 2002.

[MF06] David R. Millen and Jonathan Feinberg. "Using Social Tagging to Improve Social Navigation." In *Proc. of Workshop on the Social Navigation and Community-Based Adaptation Technologies. Conjunction with Adaptive Hypermedia and Adaptive Web-Based Systems (AH'06)*, June 2006.

[Mic] "Microsoft adCenter." `http://adcenter.microsoft.com/`.

[MLS06] Qiaozhu Mei, Chao Liu, Hang Su, and ChengXiang Zhai. "A Probabilistic Approach to Spatiotemporal Theme Pattern Mining in Weblogs." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2006.

[Mtu] "Amazon Mechanical Turk." `http://www.mturk.com`.

[NF03] Henrik Nottelmann and Norbert Fuhr. "Evaluating Different Methods of Estimating Retrieval Quality for Resource Selection." In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[OW02] Chris Olston and Jennifer Widom. "Best-Effort Cache Synchronization with Source cooperation." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2002.

[PO05] Sandeep Pandey and Christopher Olston. "User-Centric Web Crawling." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2005.

[PP07] Seung-Taek Park and David Pennock. "Applying Collaborative Filtering Techniques to Movie Search for Better Ranking and Browsing." In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2007.

[PRC03] Sandeep Pandey, Krithi Ramamritham, and Soumen Chakrabarti. "Monitoring the Dynamic Web to respond to Continuous Queries." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2003.

[QC06] Feng Qiu and Junghoo Cho. "Automatic Identification of User Interest For Personalized Search." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2006.

[QL07] Huiming Qu and Alexandros Labrinidis. "Preference-Aware Query and Update Scheduling in Web-databases." In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2007.

[QQZ06] Shouke Qin, Weining Qian, and Aoying Zhou. "Approximately Processing Multi-granularity Aggregate Queries over Data Streams." In *Proceedings of the International Conference on Data Engineering (ICDE)*, 2006.

[RD02] Matthew Richardson and Pedro Domingos. "Mining Knowledge-Sharing Sites for Viral Marketing." In *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2002.

[RDR07] Matthew Richardson, Ewa Dominowska, and Robert Ragno. "Predicting Clicks: Estimating the Clik-through Rate for New Ads." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2007.

[RL04] Daniel E. Rose and Danny Levinson. “Understanding User Goals in Web Search.” In *Proceedings of the International World-Wide Web Conference (WWW)*, 2004.

[RSSa] “RSS 2.0 Specification.” `http://blogs.law.harvard.edu/tech/rss`.

[RSSb] “RSScache.” `http://www.rsscache.com`.

[RT07] Raghu Ramakrishnan and Andrew Tomkins. “Toward a PeopleWeb.” *IEEE Computer*, **40**(8):63–72, 2007.

[SC03] Luo Si and Jamie Callan. “Relevant Document Distribution Estimation Method for Resource Selection.” In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[SDR03] Shetal Shah, Shyamshankar Dharmarajan, and Krithi Ramamritham. “An Efficient and Resilient Approach to Filtering and Disseminating Streaming Data.” In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2003.

[SHY04] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. “Adaptive Web Search Based on User Profile Constructed without any Effort from Users.” In *Proceedings of the International World-Wide Web Conference (WWW)*, 2004.

[Sut90] Richard S. Sutton. “Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming.” In *Proceedings of the International Conference on Machine Learning*, pp. 216–224, San Mateo CA, 1990. Morgan Kaufman.

[Syn] “Syndic8.com - RSS Feeds, Atom Feeds.” `http://www.syndic8.com`.

[SZ05] Xuehua Shen and ChengXiang Zhai. “Active Feedback in Ad Hoc Information Retrieval.” In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[Tec] “Technorati.” `http://www.technorati.com`.

[TIK05] Christos Tryfonopoulos, Stratos Idreos, and Manolis Koubarakis. “Publish/Subscribe Functionality in IR Environments using Structured Overlay Networks.” In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2005.

[TK98] Howard M. Taylor and Samuel Karlin. *An Introduction To Stochastic Modeling.* Academic Press, 3rd edition, 1998.

[WB08] Xiaoyuan Wu and Alvaro Bolivar. “Keyword Extraction for Contextual Advertisement.” In *Proceedings of the International World-Wide Web Conference (WWW)*, pp. 1195–1196, 2008.

[Wek] “Weka project.” `http://www.cs.waikato.ac.nz/ml/weka/`.

[WPB01] Geoffrey I. Webb, Michael J. Pazzani, and Daniel Billsus. “Machine Learning for User Modeling.” *User Modeling and User-Adapted Interaction*, **11**(1-2):19–29, 2001.

[WSY02] J.L. Wolf, M.S. Squillante, P.S. Yu, J. Sethuraman, and L. Ozsen. “Optimal Crawling Strategies for Web Search Engines.” In *Proceedings of the International World-Wide Web Conference (WWW)*, 2002.

[WZH07] Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. “Mining Correlated Bursty Topic Patterns from Coordinated Text Streams.” In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2007.

[WZY06] Xian Wu, Lei Zhang, and Yong Yu. “Exploring Social Annotations for the Semantic Web.” In *Proceedings of the International World-Wide Web Conference (WWW)*, May 2006.

[XLG03] Wei Xu, Xin Liu, and Yihong Gong. “Document Clustering Based On Non-Negative Matrix Factorization.” In *Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR)*, 2003.

[YG00] Tak Yan and Hector Garcia-Molina. “The SIFT Information Dissemination System.” *ACM Transactions on Database Systems*, **24**(4):529–565, 2000.

[YGC06] Wen-tau Yih, Joshua Goodman, and Vitor R. Carvalho. “Finding Advertising Keywords on Web Pages.” In *Proceedings of the International World-Wide Web Conference (WWW)*, 2006.

[YZ01] Qiang Yang and Henry Hanning Zhang. “Integrating Web Prefetching and Caching Using Prediction Models.” *Journal of World Wide Web*, **4**(4):299–321, 2001.

[ZBZ08] Ding Zhou, Jiang Bian, Shuyi Zheng, Hongyuan Zha, and C. Lee Giles. "Exploring Social Annotations for Information Retrieval." In *Proceedings of the International World-Wide Web Conference (WWW)*, April 2008.

[ZMK05] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. "Improving Recommendation Lists Through Topic Diversification." In *Proceedings of the International World-Wide Web Conference (WWW)*, 2005.