กรอบงานสำหรับการจัดการสารสนเทศส่วนบุคคลบนไมโครบล็อกกิงด้วยสวีบ็อค

นาย วรุตม์ สุระพัฒน์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2553

PERSONALIZED MICROBLOGGING INFORMATION MANAGEMENT FRAMEWORK BASED ON SWEBOK.

Mr.Warut Surapat

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Software Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2010

Thesis Title	PERSONALIZED MICROBLOGGING INFORMATION MANAGEMENT FRAMEWORK BASED ON SWEBOK.
By	Mr.Warut Surapat
Field of Study	Software Engineering
Thesis Advisor	Assistant Professor Nakornthip Prompoon

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.... Dean of the Faculty of Engineering

(Associate Professor Boonsom Lerdhirunwong, Dr.Ing.)

THESIS COMMITTEE

.... Chairman

(Associate Professor Wanchai Rivepiboon, Ph.D.)

Nakornthip Prompoon
.... Thesis Advisor

(Assistant Professor Nakornthip Prompoon)

.... Examiner

(Associate Professor Taratip Suwannasart, Ph.D.)

.... Examiner

(Yunyong Teng-amnuay , Ph.D.)

.... External Examiner

(Chalermsak Lertwongsatien, Ph.D.)

วรุฒม์ สุระพัฒน์ : กรอบงานสำหรับการจัดการสารสนเทศส่วนบุคคลบนไมโครบล็อกกิงด้วยสวีบ็อค. (PERSONALIZED MICROBLOGGING INFORMATION MANAGEMENT FRAMEWORK BASED ON SWEBOK) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผู้ช่วยศาสตราจารย์ นครทิพย์ พร้อมพูล , 125 หน้า

ภารกิจทางวิศวกรรมซอฟต์แวร์อย่างหนึ่งคือการหาข้อปฏิบัติ แบบจำลอง หลักการ และเครื่องมือ ซึ่งสามารถช่วยให้องค์กรลดค่าใช้จ่ายและเวลาที่ใช้ในการพัฒนาซอฟต์แวร์ได้ ไมโครบล็อกกิงเป็นหนึ่งในแหล่งข้อมูลแหล่งหนึ่งที่มีข้อมูลที่เป็นประโยชน์เหล่านี้อยู่มาก อย่างไรก็ตามข้อความบนไมโครบล็อกกิงนั้นมีความสั้น เปลี่ยนแปลงอย่างรวดเร็ว และมีเนื้อหาที่กว้าง การค้นหาข้อมูลที่เป็นประโยชน์จากไมโครบล็อกกิงจึงทำได้ไม่ง่ายนัก

ในงานวิจัยชิ้นนี้ ผู้วิจัยได้นำเสนอกรอบงานและตัววัดความเกี่ยวข้องสำหรับการจำแนกและค้นคืนข้อความจากไมโครบล็อกกิงซึ่งมีความเกี่ยวข้องในด้านวิศวกรรมซอฟต์แวร์ สวีบ็อคถูกนำมาใช้ในการสร้างตัวจำแนกข้อความจากความถี่ของคำ ข้อความจากไมโครบล็อกกิงจะถูกจำแนกหรือค้นคืนตามคะแนนที่ถูกคำนวณจากความคล้ายคลึงด้านเนื้อหาและบริบททางสังคม (social context) โดยบริบททางสังคมประกอบด้วยมุมมองด้านผู้ใช้ (user feature) และมุมมองด้านประชาคม (community feature) การทดลองเพื่อวัดประสิทธิภาพของกรอบงานนี้ถูกทำขึ้นโดยเปรียบเทียบผลลัพธ์กับวิธีการค้นคืนตามหลักการการจัดเก็บและค้นคืนสารสนเทศ โดยการวัดประสิทธิภาพของการจำแนกข้อความถูกวัดด้วยค่าเฉลี่ยฮาโมนิค และประสิทธิภาพของการค้นคืนข้อความถูกวัดด้วย weighted r-precision และ discounted cumulative gain ซึ่งจากการทดลองพบว่าประสิทธิภาพที่ได้จากการจำแนกและค้นคืนข้อความด้วยการใช้บริบททางสังคมมีมากกว่าวิธีการค้นคืนตามหลักการการจัดเก็บและค้นคืนสารสนเทศที่ระดับนัยสำคัญเท่ากับ 0.05 นอกจากนี้ผู้วิจัยได้พัฒนาเครื่องมือตามกรอบงานที่นำเสนอโดยรวมตัวคัดแยกที่สร้างจากสวีบ็อคไว้ ซึ่งช่วยให้วิศวกรซอฟต์แวร์สามารถรวบรวมข้อมูลที่เป็นประโยชน์จากไมโครบล็อกกิงได้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต วรุฒม์ สุระพัฒน์
สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่ออ.ที่ปรึกษาวิทยานิพนธ์หลัก นครทิพย์ พร้อมพูล
ปีการศึกษา 2553

5170451321: MAJOR SOFTWARE ENGINEERING

KEYWORD: MICROBLOGGING / INFORMATION FILTERING / FRAMEWORK

WARUT SURAPAT : PERSONALIZED MICROBLOGGING INFORMATION MANAGEMENT FRAMEWORK BASED ON SWEBOK.. ADVISOR : ASSISTANT PROFESSOR NAKORNTHIP PROMPOON, 125 pp.

One of the software engineering task is to find the practices, models, principles, and tools which can help the organization to reduce its cost and to save its time on software development project. Microblogging is a rich resource where these information can be found. However, the content of Microblogging message is short, rapidly changed, and diverse. Finding information in such source is not a trivial task.

In this thesis, we propose the framework and the relevance-assessing metrics for classifying and retrieving the messages from Microblogging which are related to software engineering field. The Guide to Software Engineering Body of Knowledge (SWEBOK) is selected for constructing the term-frequency-based message classifiers. The message from Microblogging is classified and retrieved according to the score computed from its content similarity to classifiers and its social context: the combination of user feature and community feature. The experiments to assess the effectiveness of the proposed framework compared to the classic Information Retrieval approach are conducted. The classification effectiveness is measured by harmonic mean and the retrieval effectiveness is measured by weighted r-precision and discounted cumulative gain. With statistical analysis, it is shown that the proposed framework is more effective than classic Information Retrieval approach in both message classification and retrieval at a level of significant 0.05. We also develop the tool according to the proposed framework that can help software engineer to collect useful information from Microblogging.

Department: Computer Engineering	Student's Signature Warut Surapat
Field of Study: Software Engineering	Advisor's Signature Nakornthip Prompoon
Academic Year: 2010	

Acknowledgements

It can not be ignored that one of the most interesting thing of human life is how it is evolved. Some people who believe in god said that the god challenges problems to those who are worth. Such problem is one of the factor that results in self development. However, it is a thing that may be hard to overcome by one's own self. My step can not be completed without the helps from many people that I would like to thank all of them for their contributions.

The first two people that I was deeply indebted the most are my mother, Mrs.Nipaporn Surapat, and my advisor, Asst. Prof. Nakornthip PromPoon. Mrs.Nipaporn unconditionally nourishes me. She always supports me with all things she has got.Whenever her children fall, she does her best to bring them up. My life was not given once at a time of birth, but also other times when I fell into the darkness. Her favor is un-returnable even if I spent my whole life for it. Asst. Prof. Nakornthip Prompoon contributes lot of her efforts to my work. She does not only shape my work, but also shapes the humanity of her adviseet. Most importantly, she gives the freedom for her student's wisdom. It is not exaggerate to say that she is the best teacher I have ever met.

My next thank goes to all thesis committee members: Associate Professor Wanchai Rivepiboon, Ph.D., Associate Professor Taratip Suwannasart, Ph.D., Yunyong Teng-amnuay , Ph.D. and Chalermsak Lertwongsatien, Ph.D, who spent their precious times for giving invaluable opinions which are not only useful, but also open many new perspectives on how to makes this work better.

I would like to forward my thank to Miss Ayumi Masuda. She is one of the people who has big impact towards my life. Even she is not by my side any more, I would like to pray for her success and wish that she would be happy.

More thanks are given to three of my most beloved friends: Mr.Chatchavan Wacharamanotham, Mr.Kittinun Vantasin and Ms.Penpicha Lakbanglam. They are always beside me and always give the best supports. There are more people that I would like to thank. Their names may not be written here, but they are, everlastingly, written in my heart.

Lastly, I would like to thank the music that brings me on. I could not be myself today and even could not live without it. It is one of the most beautiful thing on earth.

Contents

Chapter Page

Chapter Page

List of Tables

Table | Page

List of Figures

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

1.1 Research Background

One of the software engineering task is to find the practices, models, principles and tools which can help the organization to reduce its cost and to save its time on software development project. However, due to the explosive growth of technology, finding such useful information is not a trivial task. Information seeker generally gets it via searching and browsing the web. This is a time consuming process due to the reason that the information is large, diverse and is rapidly changed.

Microblogging application, such as Twitter, Jaiku and Pawnce, is one of the possible potential source where useful information about software development can be found. By letting the user posts a short text expressing their thoughts, Microblogging is considered to be one kind of the word-of-mouth communication (Jansen et al., 2009) which users tell others about their experiences, impressions or disappointments toward a particular topic. Getting information from the word-of-mouth is likely to reduce time for trial-and-errors because it was often tried or experienced by the information owner.

To receive messages from other users on Microblogging application, users must subscribe to people they are interested in. We can also consider Microblogging as a human-based News feed (Zhao and Rosson, 2009). Microblogging provides easy way to post messages by limiting the length of text and enabling various input and output channels. These characteristics motivate users to post messages often.

However, there are some problems arisen from its nature. Firstly, messages that are visible at a specific time can be quickly pushed down from the message list and they will be hard to be found later. Secondly, there are lots of unwanted messages on Microblogging as users do not post only their interests but also their daily activities, their criticism, and much more. The principles of Information Retrieval can be used to help solving these problems. However, solely based on keyword frequency it is, Information Retrieval alone may not be the best answer.

In this research, we propose the framework and metrics for classifying and retrieving the messages from Microblogging which are related to software engineering field. The Guide to Software Engineering Body of Knowledge (SWEBOK) (SWEBOK, 2004) is selected for constructing the term-frequency-based message classifiers. Messages from Microblogging stream will be classified and retrieved according to the score computed from its content similarity to classifiers and its social context (the combination of user feature and community feature). Finally, we evaluate the proposed framework by measuring its classification effectiveness with harmonic mean, and measuring its retrieval effectiveness with WPR and DCG.

1.2 Problem Statement

Given an information seeker who uses Microblogging application, how can we classify and retrieve messages according to knowledge areas defined in SWEBOK document and according to user needs?

1.3 Research Objectives

1. To design the framework to classify and retrieve messages from Microblogging application which are related to software engineering knowledge according to knowledge areas defined in SWEBOK document.
2. To develop a tool corresponding to the first objective.
3. To propose metrics for assessing importance of message in Microblogging application.

1.4 Research Scopes

1. Twitter is selected as the candidate of Microblogging application.
2. This research focuses only the messages written in English.
3. The scalability and the performance of the tool are not taken in the consideration.
4. The version of the SWEBOK used in this research is SWEBOK 2004.
5. The input of the classifier construction process is the document in the text format (.txt). This input is obtained from the content of SWEBOK document.

6. The effectiveness evaluation of the framework will be divided into two parts. The first part is the classification evaluation which is evaluated by the harmonic mean. The second part is the retrieval evaluation which is evaluated by the weighted r-precision and discounted cumulative gain at various document cutoff.

7. To evaluate the framework, the Microblogging messages corpus will be created by collecting the messages based on the simulated user network.

8. The messages in the corpus will be divided into two groups. The first group is used for profile construction while another is used for classification evaluation.

9. For retrieval effectiveness evaluation, 50 queries will be used.

10. In this research, we simulate the user network by creating the user on Twitter. We manually select other Microblogging members by using ten knowledge area titles as the queries on Twitter in order to acquire the members corresponding to each knowledge area. The profiles and the recent messages of each member in each result will be scrutinized. The particular member will be selected if he/she related to the particular knowledge area and had at least two subscription relations with the selected members.

11. The tool will be developed as the stand alone desktop application and contains the classifiers which are created from SWEBOK 2004. This tool is built on top of the Twitter API. Its fundamental functions are to let the user posts, browses and searches the messages, to let the user manages his friends list and to let the user manages the classifiers.

1.5 Research Contributions

This research will give the contributions on the following points.

1. The framework for classifying and retrieving the messages from Microblogging according to the knowledge areas define in the SWEBOK are provided.

2. The sets of metric which assess the relevance of the message, not only by it content, but also the owner's interest and the impact to the community, are provided.

3. The tool for classifying and retrieving the useful messages from Microblogging are developed.

4. On the perspective of software engineering, the software engineer can use the tool to collect the useful information such as the technology News, lessons learned and solutions toward software development. The tool can also be applied to some special purposes such as the bug tracking or the user satisfaction evaluation by fetching the related documents instead of SWEBOK and adding some specific purpose modules to work after the classification phase.
5. The proposed framework can be applied to other domain contents depending on the documents used for constructing the classifiers.

1.6 Research Procedure

1. Study the related knowledge which includes
   (a) The knowledge on Information Retrieval.
   (b) The characteristics, the usages, the structures and the benefit of Microblogging technology.
   (c) The knowledge on Social Network Analysis.
   (d) The guide to Software Engineering Body of Knowledge.
2. Define the terminology that will be used within this research.
3. Design the proposed approaches.
   (a) Design the classifier construction process.
   (b) Design the data collection process.
   (c) Design the algorithms for message classification.
   (d) Design the algorithms for classifier expansion process.
   (e) Design the necessary data structures.
4. Determine the evaluation process and metrics.
5. Develop the tool.
   (a) Design the system functions, the UI and the database schema.
   (b) Implement the tool.
6. Conduct the experiment and evaluate the approach.
7. Summarize the result and document the thesis.

1.7 Thesis Outline

In the next chapter, the background knowledges which includes the basic concept of information retrieval, SWEBOK, Microblogging and Twitter are described. It also includes some of related wokrs. In chapter 3, the approach of the framework over the message classification and retrieval is described in details. We then show the results of the evaluation experiment together with statistical analysis and discussions in chapter 4. The implementation concept of the framework and the tool are given in chapter 5. Finally, the conclusion, limitations, and future work of this research are summarized in chapter 6.

CHAPTER II

BACKGROUND KNOWLEDGES AND RELATED WORKS

In this chapter we describe the background knowledge and the researches that relate to our work.

2.1 Information Storage and Retrieval

Information Storage and Retrieval principles mainly concerns with how we can store the information and how we can retrieve it in the way that the users will be satisfied. It has three crucial processes: Storage process, Retrieval process and Evaluation process. The Details of these processes will be described within this section.

2.1.1 The Storage Process

2.1.1.1 Automatic Indexing Process

In Information Retrieval (IR) system, after the document is stored in the system, the document representation is needed to be created. This is known as the Automatic Indexing process. With this process, the document will be abridged into the set of candidate keywords called index. The index makes the search less expensive and feasible to perform when there is a lot of information available on the system as it provides the direct access to the desired documents. To determine which term is appropriate for using as index, there are the operations to be performed as depicted in Figure 2.1.

1. **Splitting the document into tokens.** In this step,the space, the new line symbols and other marks which are not located between characters will be used as the delimiters. These delimiters will be used to split the document into tokens.
2. **Eliminating the tokens which are considered as words in Stoplists.** The Stoplists (William B. Frakes, 1992) are the terms that appear often in most documents such as ‘a’, ‘the’, ‘of’, and ‘with’. These tokens are useless to be used as index because of the lack of discrimination capability among the document collections.

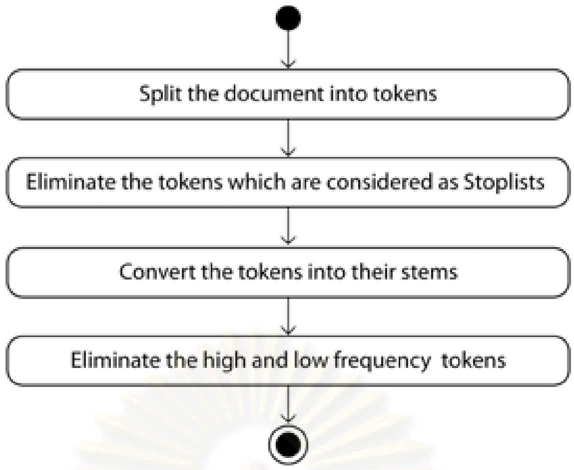


Figure 2.1: Activity diagram of automatic indexing process.

3. **Converting the tokens into their stems.** The tokens we can extract from the document are in various forms such as the plural form, the past form and the past participle form. However, the different forms give the same meaning. Therefore, the tokens are needed to be converted into the stem in order to be recognized as the same word. This method is called Stemming. There are many algorithms available, for instance, Porter's algorithm and Snowball algorithm. The Porter's algorithm (M.F. Porter and Robertson, 1980) will be used in this research.

4. **Eliminating the high and low frequency tokens.** According to (Salton and McGill, 1983), the terms with too high and too low frequency are not the good discriminators. Therefore, we need to remove them.

2.1.1.2 Automatic Term Weighting Process

After the index terms were acquired, they must be weighted. There are many alternative ways to weight the terms, for example, by using inverted document frequency, by using signal-noise ratio and by using the term discrimination value. In this research, we selected the inverted document frequency (IDF) (Baeza-Yates and Ribeiro-Neto, 1999) which can be computed using the equation 2.1. Together with the normalize frequency f_i, j of term i, which can be computed using the equation 2.2, the weight of term i in document j, $w_{i,j}$, can be computed by multiplying equation 2.1 and 2.2 as shown in equation 2.3.

$$idf(i) = log_2(n) - log_2(docfreq(i)) + 1 \tag{2.1}$$

Where idf_i is the inverse document frequency of term i, n is the number of document in the collection, and $docfreq_i$ is the number of document containing term i.

$$tf(i,j) = \frac{freq(i,j)}{max_l freq(l,j)} \tag{2.2}$$

Where $tf(i,j)$ is the normalize frequency, $freq(i,j)$ is the frequency of term i in document j, and $max_l freq(l,j)$ is the maximum frequency of all terms in the document j.

$$w(i,j) = tf(i,j) \cdot idf(i) \tag{2.3}$$

The term weight will play the important role on comparing the similarity between two documents. If the term i had high frequency in document j, its $w_{i,j}$ would increase. However, if most of documents in the collection also contain term i, the $w_{i,j}$ will be decreased by the value of idf_i. The idf_i value will increase if the term i appears in a few documents, and will decrease if the term i appear in more documents, which can be implied that the term i is a common term.

2.1.2 The Retrieval Process

The retrieval process mainly focuses on how to retrieve the stored information according to the user needs. Unlikely to the Data Retrieval which the retrieved results contain only matched records, the IR system will show to the users the results which are relevance to the user query. In order to do so, the similarity between user query and documents in the collection must be computed. After the similarity values are acquired, they will be used for ranking the result. Therefore, the users can select the documents which are similar to their intentions. This process is depicted by Figure 2.2

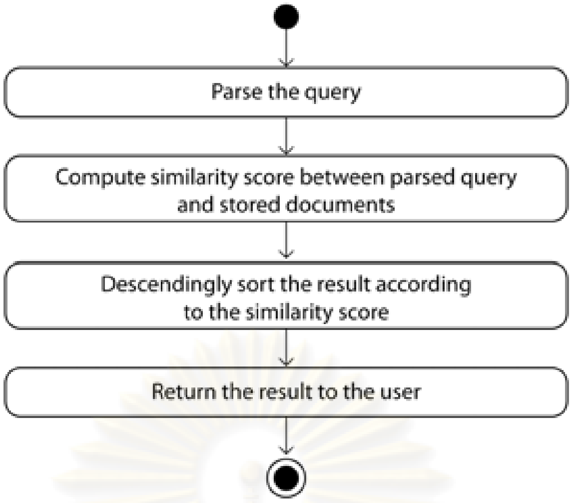


Figure 2.2: Activity diagram of retrieval process.

There are various choices on computing the similarity. Among them, we decided to use the Cosine similarity (Salton and McGill, 1983) which can be computed using the equation 2.4.

$$cosine(query_i, doc_j) = \frac{\sum_{k=1}^{t} term_{i,k} \cdot term_{j,k}}{\sum_{k=1}^{t} term_{i,k}^2 \cdot \sum_{k=1}^{t} term_{j,k}^2} \quad (2.4)$$

Where $term_{i,k}$ is the weight of term k in query i, and $term_{j,k}$ is the weight of the term k in document j.

2.1.3 The Evaluation Process

For every IR application, the general way to measure the retrieval effectiveness is to assess recall and precision (Salton and McGill, 1983). recall is the ratio between the number of relevance document that is retrieved and the total number of relevance document. It reflects how well the system can discover the relevance document. In this research, we want to investigate the improvement of the system that implements the social context in term of precision. Therefore, recall is not the metric that should be used.

Instead, we use precision which reflects how precise the retrieval capability is according to the user need. It can be computed by dividing the number of relevance document that is retrieved with the total number of retrieved document as shown in equation 4.1.

$$precision = \frac{retrel}{retrel + retnrel} \tag{2.5}$$

Where $retrel$ is the number of relevance document that is retrieved and $retnrel$ is the number of the non-relevance document that is retrieved.

Generally, precision is often used in term of r-precision (precision@r) which considers the precision for the top r items. For example, precision@5 measures the precision of top five retrieved items. It is defined as equation 2.6

$$precision@r = \frac{retrel}{r} \tag{2.6}$$

In this research, we selected r-precision family metrics for retrieval evaluation which more details are described in section 4.1.3.2.

2.2 The guide to Software Engineering Body of Knowledge 2004

The guide to Software Engineering Body of Knowledge 2004 (SWEBOK, 2004), as known as SWEBOK, is the standard document published by IEEE which its main objectives are

1. To promote a static perspective of software engineering.
2. To define the scope and boundary of software engineering with respect to other disciplines.
3. To characterize the content of software engineering discipline.
4. To provide the topical access to the Software Engineering Body of Knowledge.
5. To support certification and licensing.

This document was written in the non-technology dependent manner. It describes the main important concepts and provides the access to the necessary literature. SWEBOK categorizes the knowledge of software engineering into ten knowledge areas which are

1. Software Requirement
2. Software Design
3. Software Construction
4. Software Testing
5. Software Maintenance
6. Software Configuration Management
7. Software Engineering Management
8. Software Engineering Process
9. Software Engineering Tools and Methods
10. Software Quality

Each knowledge area is also divided into sub-areas. As the technology in this era grows very fast, this document can serve as only the fundamental guide to specific knowledge area. Software engineers usually need to find more practical approaches on implementations and should gain knowledge in other disciplines such as Management, Computer Science and Computer Engineering. In this research, we use the details of each knowledge area defined in this document to create the Microblogging message classifiers. We expected that, by using those contents as basis, we can gather the useful information from Microblogging application which can support the software development.

2.3 Microblogging and Twitter

2.3.1 Microblogging

Microblogging, a kind of Online Social Networks, has emerged and grown with an amazing rate. Its main objective is to let the users post the messages with less effort. These message may express the owners' ideas, the problems they are facing, the solution or the interesting articles. In addition, we found that Microblogging is applied to some uses such as the weather report service and the traffic report service. Most of Microblogging applications share the common characteristics as below.

1. **Limited-length message.** This characteristic helps the users to update their statuses with minimal effort, unlikely to the blog which often requires the users to spend more time and work. As a result, the Microblogging users tend to update their statuses more often.

2. **Various input and output channels.** Most of Microblogging applications provide a variety of input and output methods and make them available over various devices. For example, the message can be posted directly from the web site or the mobile device. The users can view the messages on the web, or get the update via the Really Simple Syndication (RSS). The developers are provided with the application programming interface (API) to get the capability to use the Microblogging service within their products.

3. **Wide range of information in many domains.** Due to the large number of users and their diversities, the content of messages over Microblogging covers many topics including the users daily activities. As a result, it contains a large portion of noise.

4. **Broadcasting manner.** Microblogging can be viewed as another type of SMS (Short Message Service), however, the main difference is that Microblogging is published in the broadcasting manner. Unless the owner decided to protect his/her updates, they can be viewed and accessed by public.

On Microblogging application, one can get the update from others after he/she subscribed to the people of interest. The subscription is the unidirectional relation. Suppose we have two users: u_a and u_b. If u_a subscribed to u_b, he/she would see all updates from u_b. On the other hand, u_b will not see any updates from u_a unless u_b subscribes him/her back.

2.3.2 Twitter

There are many online applications that implement the Microblogging concept. Among them, Twitter is the most well known. The survey from Nielsen, the marketing analysis firm, stated that its year-over-year growth from February 2008 to February 2009 hits 1382 percent (Growth of Twitter, 2009) and Twitter has totally 44.5 Million users (Wikipedia, 2009).

In our research, we selected Twitter as the candidate of Microblogging because of the following reasons. First, Twitter is widely used Microblogging application. It gains the largest number of users compared to others. Second, the application supports some syntax to extend the usage such as the use of @ symbol to address the conversational target. Third, the application is equipped with the best full-function-able API and is supported by many available wrapper libraries in many programming languages. Fourth, most of the researches done under this topic often select Twitter. For instance, the study about the role of Microblogging in the informal communication at work (Zhao and Rosson, 2009), the study about the usage and communities on Microblogging application by Akshay Java et al. (Akshay Java, 2007), the study of brand sentiment mining over Microblogging (Jansen et al., 2009), and the study of the use of Microblogging on the live event (Shamma et al., 2009).

On Twitter, we say that we are following someone if we subscribed to him. We say that we have followers if there was someone who decides to follow us. We also call the message on Twitter as 'Tweets'. The example of Twitter page containing the messages from various users is shown in Figure 2.3. In this figure, the number of following and followers are shown (labeled with (1) and (2)). The users can post the message by submitting the text through the text box labeled with (3) and can view the messages in the area marked with (4). The structure of the Tweets is shown in Figure 2.4. The label (1) in this figure is the Tweets' owner picture and the label (4) is his/her user name. The label (2) is the message content which may contain the link to external web resource (labeled as (5)). The posted time and client name are labeled as (3). The client here means the application which the users use to access the Twitter service, for example Echofon, TweetDroid and Twhirl. Some examples of useful Tweets related to software development field are shown in Figure 2.5.

Akshay Java et al. (Akshay Java, 2007) conducted the research on the user types and user intentions on Twitter using the dataset collected within two months period with the size of approximately one Million records. They pointed out that the user intentions over Twitter can be classified into four categories according to the number of incoming links and outgoing links:

1. **Daily chatter.** The users with this intention use Twitter to update their daily activities such as what are they doing at specific time.
2. **Conversations.** The users with this intention use Twitter as a tool for conversation by directing the message to target using @ symbol.
3. **Sharing information.** The users with this intention use Twitter to collect and post the link in which they are interested. The URL shorten service like Tinyurl, Bit.ty, and others are used for shortening the URL to fit the limited length space. The work from Huges et al.

Figure 2.3: Example of Twitter page and its components.

(Amanda Lee Hughes, 2009) found that the number of the messages in their collection that contain the link is about 24.57 percent, 11 percent increased from the research of Akshay Java et al. (Akshay Java, 2007)

4. **Reporting news.** The users with this intention use Twitter to report the news or the live-event they are participating.

The user of Twitter can be classified into 3 categories as below.

1. **Information source** is the users who post the messages often and have a lot of subscriber. However, the number of the people whom will be subscribed back by this users is less. This category include both real human and the automated tools.
2. **Friends** is the common users who use Twitter to keep listening to their friends' activities.
3. **Information seeker** is the users who rarely post the messages but often listen to others.

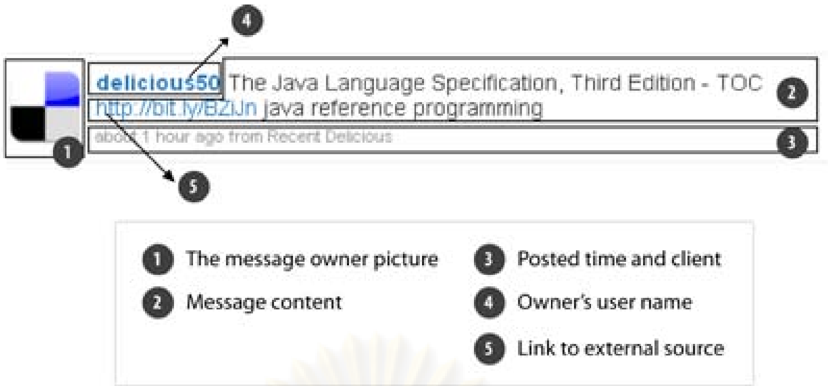


Figure 2.4: Structure of Twitter message.

Figure 2.5: Example of useful Twitter messages.

2.4 Related Works

2.4.1 Sifting Micro-blogging Stream for Events of User Interest (Maxim Grinev, 2009)

This research proposed the method on how to extract the event information from the Microblogging. From the observation, the authors stated that for any subjects of interest, there exists the period where the subject is mentioned by many messages than normal, and those messages are not much different from each others. Therefore, the method for detecting the interested event can be done using the frequency analysis. The messages on the Microblogging stream is examined and will be counted if they matched to the search query entered by the user. With the frequency of messages counted at many points of time, the peak periods can be detected. All the messages within the peak periods will be clustered by their similarity values. The central messages of the most dense cluster will be selected as the message that best describes the event.

Compared to our research that tries to capture the useful information, there might be some peak periods where users discuss about new technology. However, this requires the search query to be available at first. It is different to our research that we do not have the query at the beginning Therefore, to detect the trending topics, we propose the use of term score metric to judge whether the term should be beneficial enough to be added to the classifiers.

2.4.2 Using Twitter to Recommend Real-Time Topical News (Owen Phelan, 2009)

Using the fact that the available information on Microblogging can reflect the interest in real-time. This research tries to create the real-time news recommendation engine by analyzing user's submitted RSS and the content feed of Twitter. The idea of the analysis is to find the term co-occurrence between those 2 sources. Those terms found in both sources will be used as the query to retrieve the article from the database. The retrieved result will be re-ranked based on the score which can be computed by the summation of the tf-idf of each term in each document. Threes recommendation strategies are provided to the users. The first strategy, Content-Rank, is to use only the RSS as the source. Next, Public-Rank strategy, is to use the RSS together with the Twitter public time line feed. The last strategy, Friends-Rank, uses the RSS with the Twitter feed from friends' time line. Even there is the conflict between the result from the experiment and from questionnaire on the preference of Public-Rank and Friends-Rank strategies, the study stated that using Twitter as a source for recommendation is preferable by the users, and those who have more friends should benefit more.

Even the objective of this research is different from ours, the intention of the process is quite familiar. However, the disadvantage of the method proposed in this research is that the recommendation is too dynamic. In the case that the user interests are not changed over time, this approach may not serve the good matched articles. Opposing to this approach, our research removes this drawback as we select the base static document and extend its capability by adding new terms if they were considered to be important enough. Thus, our approach can filter the message in both static and dynamic manners.

2.4.3 Micro-blogging as Online Word of Mouth Branding (Jansen et al., 2009)

In commercial, the word-of-mouth is the process of giving the information about a particular product or topic from one person to another. It is considered as a powerful type of communication which can strongly influence the customer as the word-of-mouth is based on the social trust.

The authors of this research mentioned that Microblogging is a potential channel for the online word-of-mouth marketing. Therefore, they want to investigate how the word-of-mouth over the Microblogging application is by scrutinizing the expression of the brand attitudes. The research was conducted by using Summize tool which is the Microblogging searching service to monitor the sentiments of the 50 selected brands that were changed over 13 weeks period. Approximately, 140K messages over Twitter were analyzed. The interesting point in this research is that only 650 messages mentioned about the selected brands. The reason for this might be that the message collection was done over all of the users available on Twitter. As a result, the possibility that the selected brand will be mentioned is very low. Although a small number of messages could be gathered, this research had shown that analyzing these messages is feasible and may lead to the useful result.

Regarding to our work, not only this research confirms to us the usefulness of the message over Microblogging, it also prompts to us the problem we need to consider. We decide to focus on the group of users who are likely to share the same interest instead of gathering the message from all the users and propose the community feature metric for this sake. Collecting the message from the group, not only we can limit the scope of user, we can also get more relevance information.

2.4.4 Efficient Top-k Querying over Social-Tagging Networks (Ralf Schenkel, 2008)

Social tagging is the application that lets the users in community annotate the interesting documents using their own keywords called tags. The document recommendation can be made between the users who have relations to each other. In addition, the tag can be used for searching the document. However, the existing researches mentioned only the uses of the tag on searching without considering the relation between the searcher and the user who owns or annotates the documents. Another problem is that the rapid user growth produces an immense number of document. Therefore, the higher system efficiency and scalability is needed.

This research tried to solve these problems by proposing model and algorithm for social searching and ranking. The social expansion and semantic expansion were introduced. The social expansion is the most interesting part which is most related to our research. One characteristic of the social tagging system is that a document can be tagged by one or more users. Therefore, there are the relations between the entities in this system; between the user, the document and the tags. To score the term in the document over social tagging application, the authors proposed the social scoring model, which uses the social frequency in the calculation instead of the legacy term frequency. The social frequency of tag t over document d is the summation of the number of time tag t is used to annotate document d weighted by the similarity and the strength of relation of the user who submitted the query and the user who tagged the document. The social frequency is high when lot of users who are closer to the query submitter has used the tag t to annotate document d, and the similarity between that user and the query submitter is high.

There are the differences between the Microblogging and the social tagging application. Firstly, the relation between user and document on the Microblogging is in the one-to-one manner; the document is owned by one user. Meanwhile, the relation between user and document on the social tagging is many-to-many; the document is tagged by one or more users. Secondly, the social tagging can be viewed as a process of manual indexing, while the documents over Microblogging are parsed to the automatic indexing process. These differences make the social frequency unusable. Therefore, we proposed a new set of metrics which is more suitable for use in the Microblogging environment. With the different point of view, we defined the community feature, and use it in the flexible manner that it can be enabled, disabled, or partially weighted to fit the user intention.

2.4.5 A Proposal for a Semantic Intelligent Document Repository Architecture (Rodriguez et al., 2009)

In this research, the authors mentioned about the problem of the research article disorganization due to the multiple research repositories. The architecture for classifying the document which focuses on the software engineering domain is presented. Instead of using only the keyword based classification, the ontology extracted from SWEBOK is also used. The documents are parsed to the extraction process which produces the document descriptions in RDF and OWL formats. This document descriptions are used to compare with the SWEBOK ontology in order to classify the document.

Compared to our research, SWEBOK is selected for classifier construction, however, to classify the message, we used the benefit from the network structure where the trustfulness and interest sharing can be implied. The ontology comparison is discarded because the term relation is hard to be extracted due to the equally terms distribution in the message over Microblogging which is the result of the limited length message characteristic.

CHAPTER III

APPROACH

In this chapter, approach for information classification, storage, and retrieval are described. We begin the chapter with the glossary to ensure that the readers can consistently understand what we want to convey throughout the document. Next, the classic IR approach for message classification and message retrieval is described. The overview of social context approach is briefed before the detail in each part is described together with the proposal of new metrics. Finally, we conclude the differences of our approach to the classical IR approach and the list of proposed metrics. Please be noted that the tool development which is one of the research objectives will be described in Chapter 5.

3.1 Glossary

To ensure that the content of this document will be understood in the same way, important terms are defined as shown in Table 3.1.

Table 3.1: Important term definitions

Term	Definition
Message, Tweet	The limited-length message that is published by the user or his friends.
Information Seeker	The user who wishes to classify software engineering related messages from those available on Twitter message stream.
User	The person who uses Twitter service.
Author	The owner of a particular message.
Relation	The connection between two users, either to follow and to be followed.
Follow	The subscription from one Twitter user to another over Twitter application.
Follower	The user who follows another user.
Friends, Followee	The user who is followed by another user.
Timeline	The stream of messages from whom the user subscribes to.
Personal Information	The information of a particular user which includes name, short biography, number of followers and number of followees.
User Network	The graph that presents relations between a particular user and his friends (and also the relation between each friends).
Social Context	The combination of user feature and community feature.

3.2 Classic IR Approach

In this section, the detail of classic IR approach for message classification and message retrieval will be roughly described to fill the reader's background. The classic IR approach for message classification and message retrieval will be used as baseline in our experiment which its detail will be given in Chapter 4.

3.2.1 Message Classification

To classify a message according to software engineering, firstly, classifiers is needed to be constructed from SWEBOK. As the classifier construction is one of processes in our work, we skip its detail here and describe such detail later in Section 3.4. After the classifiers is acquired, they are used to compare their similarity to a message's content. Given the message m and the classifier c, the similarity between m and c is defined as equation 3.1.

$$ContentSim(m, c) = \frac{\sum_{t \in T_m} (w_{t,m} \cdot w_{t,c})}{\sqrt{\sum_{t \in T_m} w_{t,m}^2 \cdot \sum_{t \in T_c} w_{t,c}^2}} \tag{3.1}$$

Where T_m is the set of terms in message m, T_c is the set of terms in the classifier c, $w_{t,m}$ and $w_{t,c}$ are the weight of term $t \in T_m$ and the weight of term $t \in T_c$ respectively.

If $ContentSim$ between the message m and the classifier c exceeded the predefined threshold, m is decided as a member of the category corresponding to c.

3.2.2 Message Retrieval

To retrieve messages according to a query, the similarity between a query and all messages are computed according to equation 3.1. After the computation is done, the messages are sorted with their similarity scores in descending order and are returned to user.

3.2.3 Issues

The classic IR approach towards message over Twitter is not effective due to the following issues

1. The message over Twitter has limited length and the terms in the message are likely to occur only once.
2. The message over Twitter contains lot of specific terms. However, the classifier created from SWEBOK contains lot of broad terms.

We overcome these issues by using the social context which take the user feature and community feature in consideration. The details of our approach is described in the coming section.

3.3 Social Context Approach : Overview

In this research, we focus on how to store, classify, and retrieve the messages from Microblogging application according to the software engineering knowledge, and the personal interests of the information seeker u. Mainly, the approach of this research is divided into four phases, as shown in Figure 3.1, which are classifier construction phase, user data preparation phase, classification phase, and retrieval phase respectively. Rough activity descriptions of each phase are described as follows.

1. **Classifier Construction Phase**
   (a) Each knowledge area in SWEBOK is mapped into one document. Totally, ten documents are created.
   (b) The classifiers are created by parsing ten documents to the automatic indexing process and term weighting process.
2. **User Data Preparation Phase**
   (a) The user network of the information seeker and his friends is construction.
   (b) The recent messages of all user are retrieved and used for profile construction.

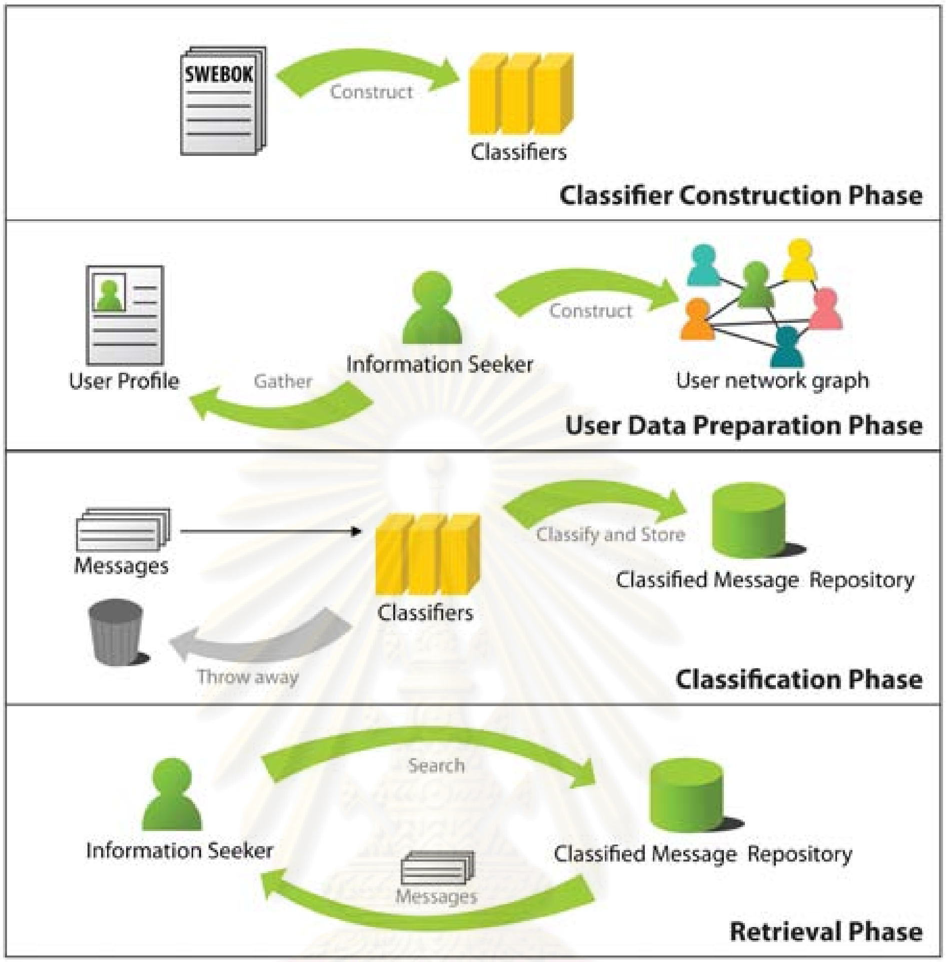


Figure 3.1: Approach overview.

3. **Classification Phase**

    (a) The messages are retrieved and sent to the classifiers.

    (b) The messages are classified. Those related to software engineering are kept.

    (c) The new terms from the classified messages are collected and are used to compute term weight in order to add them to the classifiers.

4. **Retrieval Phase**

    (a) The information seeker submits the query.

    (b) The messages are searched according to the similarity and are returned to the information seeker.

3.4 Social Context Approach : Classifier Construction Phase

Microblogging is known as a real-time information source. However, the number of messages on the stream is large and contains un-useful messages. Some Twitter users have to spend more time reading and filtering them manually. To solve this problem, the classifiers are needed so that the automatic filtering can be done. As we focus on software engineering related content, we select SWEBOK as the source for classifier construction. The classifier will help us on filtering by assess the textual similarity between itself and a message's content.

To construct the classifiers, firstly, the knowledge areas which are divided as SWEBOK chapters are divided into documents. All documents are added to the knowledge area collection which is parsed to the automatic indexing process as described in section 2.1.1.1. However, the terms which have high frequency and are not the member of Stoplists set will not be eliminated. Totally, ten sets of index are created. The classifier can be obtained by parsing these index sets to the term weighting process. Finally, the collection of classifier $C = \{c_0, c_1, ..c_9\}$ is returned as the result. This process is depicted in Figure 3.2

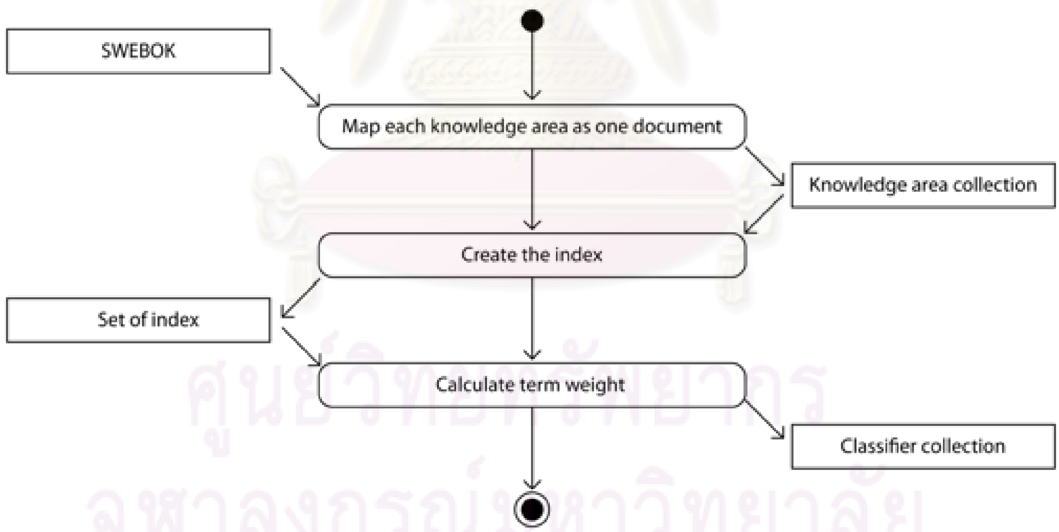


Figure 3.2: Activity diagram of classifier construction process.

3.5 Social Context Approach : User Data Preparation Phase

Using textual similarity to find document that matches the user need is effective in most of IR application. However, it is not sufficient for Microblogging messages which often have the limited length. Our approach regarding to this problem is to invoke the social context which includes the user profile and the user relationships instead of using only the textual similarity.

Let $S = \{s_0, s_1, ..., s_j\}$ be the set of all subscription relations where $s_j = f_a \rightarrow f_b$ be the subscription relation from user f_a to user f_b and $S_{f_a} = \{s_k | s_k = f_a \rightarrow f_k\}$ be the set of all subscriptions from f_a. The distance from f_a to f_b, denoted by $Distance(f_a, f_b)$, is the minimum number of edges between f_a and f_b. In our research, as information seeker u is focused, to limit the scope of classification, the user network G which represents users and relations among them is defined as $G = (V, E)$. Let f be the user, $F = \{f | Distance(u, f) \leq G_\delta\}$ be set of users of interest where G_δ is the maximum distance measured from u, V is the set of vertex defined as $V = \{u \cup F\}$ and $E = \{s_i \in S\}$ is the set of edge. It is favorable to set G_δ as a small number as the closer users are more likely to share common interest (Bernstein et al., 2010), (Sarwar et al., 2002). The user network G can be constructed using algorithm 1 and 2.

Algorithm 1 ConstructUserNetwork

Require: Information Seeker u, maximum distance G_δ
  $distance = 0$
  $V \leftarrow \{u\}$
  $E \leftarrow \{\}$
  while $distance \leq G_\delta$ **do**
    $ExpandNetwork(V, E, distance)$
    $distance \leftarrow distance + 1$
  end while
  $G \leftarrow (V, E)$
  return G

After the user network is constructed, recent messages of every user $f \in F$ are retrieved. They will be parsed to the automatic indexing and term weighting process in order to create the profile p_f which represents the user interests.

According to Twitter, the information seeker can receive messages from users who he/she follows. Therefore, in this research, we are interested only in uni-directional relationship. This means that, we are interested in the subscription from the information seeker to other friends (or from one friend to other friends) without considering the subscriptions from those friends to the information seeker.

Algorithm 2 ExpandNetwork

Require: User list V, subscription list E, distance D
for all $u_i \in V$ **do**
  if $Distance(u, u_i) == D$ **then**
    $V' \leftarrow retrieve\_user(u_i)$ {get the users}
    $E' \leftarrow retrieve\_relation(u_i)$ {get the relations}
    for all $u_j \in V'$ **do**
      if $u_j \notin V$ **then**
        append u_j to V
      end if
    end for
    for all $s_k \in E'$ **do**
      if $s_k \notin E$ **then**
        append s_k to E
      end if
    end for
  end if
end for

3.6 Social Context Approach : Classification Phase

Given the message m published by user f, the classification processes to determine the relevance of m according to ten software engineering knowledge areas using the classifiers in classifier collection C are described in this section. The overview of this phase which consists of two main activities: classifying the message and expanding the classifiers, is depicted by Figure 3.3.

3.6.1 Message Classification

The objective of message classification phase is to classify a message according to the knowledge areas defined in SWEBOK. If a message was relevance to one knowledge area or more, it would be kept and used for expanding the classifiers.

To assess message relevance, we define three features of message; content feature, user feature, and community feature. The combination of user feature and community feature is defined as 'social context'. The reason behind social context is that, as Microblogging message is short and its content is diverse, solely assessing its relevance from textual similarity may not sufficient enough. Instead, author of the message should be considered. Firstly, if author has profile that is similar to the interest of information seeker, the message published by such author should have higher chance to be relevance. This is the idea of user feature. Next, author that is considered 'important', by other users who share common interest with information seeker, should also have

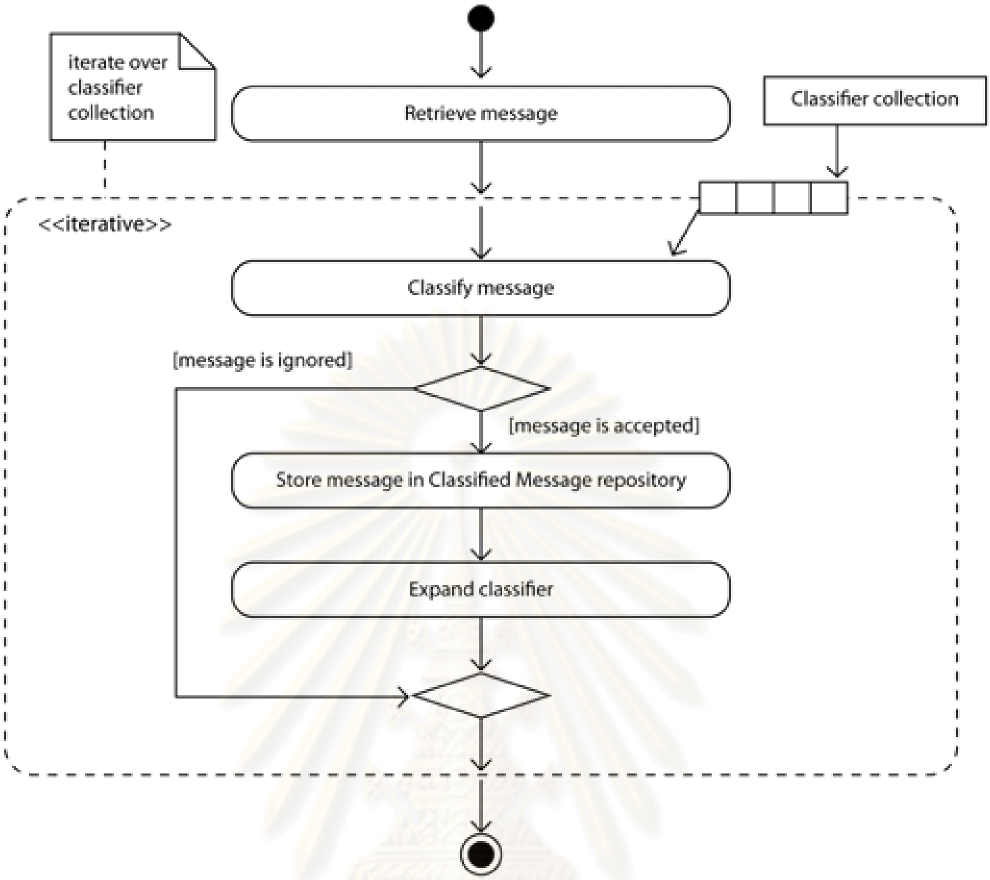


Figure 3.3: Activity diagram of classification phase.

higher chance to publish relevance message, too. The impact of author, i.e., the importance of author, can be assessed from the relations he has. And this is the idea of community feature.

The message classification process is shown in Figure 3.4. The message m which may contain the link to external resource l is firstly parsed to the indexing process in order to get its representation. Then, the scores of each feature is calculated. For convenience, user f who publishes message m will be referred as m's author.

3.6.1.1 Content Feature

The basic feature that can help message classification is its content. In the same way as the classic IR approach, we measure a message relevance according to its similarity to a classifier as defined in equation 3.1.

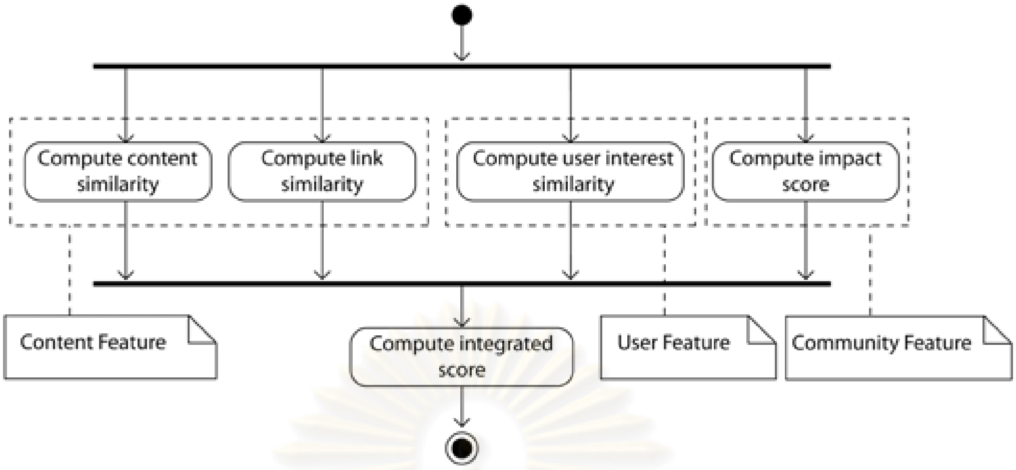


Figure 3.4: Activity diagram of message classification process.

In addition, a message may contain a link to external resource l. If such resource exists, it will be retrieved and be compared to the classifier. We define the link similarity between l and the classifier c as equation 3.2.

$$LinkSim(l, c) = \frac{\sum_{t \in T_l} (w_{t,l} \cdot w_{t,c})}{\sqrt{\sum_{t \in T_l} w_{t,l}^2 \cdot \sum_{t \in T_c} w_{t,c}^2}} \tag{3.2}$$

Where T_l is the set of terms in external resource l, T_c is the set of terms in the classifier c, $w_{t,l}$ and $w_{t,c}$ are the weight of term $t \in T_l$ and the weight of term $t \in T_c$ respectively.

3.6.1.2 User Feature

The author should be considered as one factor for deciding whether the published messages have a possibility to fall under a particular category or not. This possibility can be determined from the similarity between author's interests and classifiers. To understand author interests, profile – messages that the author published in the past– is investigated.

Let p_f be the profile of author f which is constructed in the user data preparation phase. p_f is the vector which its members are the weights of term. Each message is treated as a document. The author interest according to a given classifier c is determined from the similarity of p_f and c

which is defined as equation 3.3.

$$UserInterestSim(p_f, c) = \frac{\sum\limits_{t \in T_{p_f}} (w_{t,p_f} . w_{t,c})}{\sqrt{\sum\limits_{t \in T_{p_f}} w_{t,p_f}^2 . \sum\limits_{t \in T_c} w_{t,c}^2}} \tag{3.3}$$

Where T_{p_f} is the set of terms in user profile p_f, T_c is the set of terms in the classifier c, w_{t,p_f} and $w_{t,c}$ are the weight of term $t \in T_{p_f}$ and the weight of term $t \in T_c$ respectively.

3.6.1.3 Community Feature

In addition to message's user feature, author who has higher impact, i.e., author who is considered to be important, should have higher chance that his messages will be relevance. As we focus on the personalized classification, the impact of the user are assessed in two perspectives:

1. The author impact toward all users $f \in F$.
2. The author impact the information seeker u.

Basically, the impact can be calculated based on link structure. However, link structure on the user network graph only reflects overall impact, i.e., without concerning topic of interest. It does not reflect the impact on a particular topic of interest. For instance, given the user network of 10 members as shown in Figure 3.5. The color in this figure indicates the group of interest: the members of darker color group share the interest in Software Design topic, while the members of lighter color group's share the interest in Software Configuration topic. User A has five incoming links. We could say that he has the highest overall impact. However, his impact on Software Design topic is low as he has only one incoming link from users who are interested in Software Design. On the other hand, User B has only three incoming links. His overall impact is lower than User A, but his impact on Software Design is higher as he has 2 incoming links from the users who share the same interest in this topic.

To get the impact according to topic of interest, let $G'_c = \{V', E'\}$ be the reduced user network graph. Given the user network graph G and the classifier c, V' can be obtained by two ways:

Figure 3.5: Example of overall impact and impact on a particular topic of interest.

1. Removing all user f whose $UserInterestSim(p_f, c) < G'_\gamma$ where G'_γ is the predefined threshold as shown in Figure 3.6(a). Using this method, the author who has rarely or never published the messages related to c will be removed from G.

2. Ranking $UserInterestSim(p_f, c)$ and removing user whose rank is lower than G'_r where G'_r is the cutoff position as shown in Figure 3.6(b). Using this method,the size of the user in G'_c will be fixed, and the chance that the messages from author who has rarely or never published about c will be increased.

After V' is obtained, E' can be acquired by removing every subscription $s_k \in E$ of the users who are not in V'.

The impact of the author f toward all users in G' is defined as equation 3.4.

$$NS_{G'_c}(f) = \frac{Number\ Of\ Subscriber(f)}{max\ Number\ Of\ Subscriber} \tag{3.4}$$

Where $Number\ Of\ Subscriber(f)$ is the number of the users who subscribe to f in G'_c.

The impact of author f on u can be determined from the similarity between f's interests and u's interests. We compare interest between them using similarity of their relations. If both of them have some common followees, we could imply that they may share same interest. The

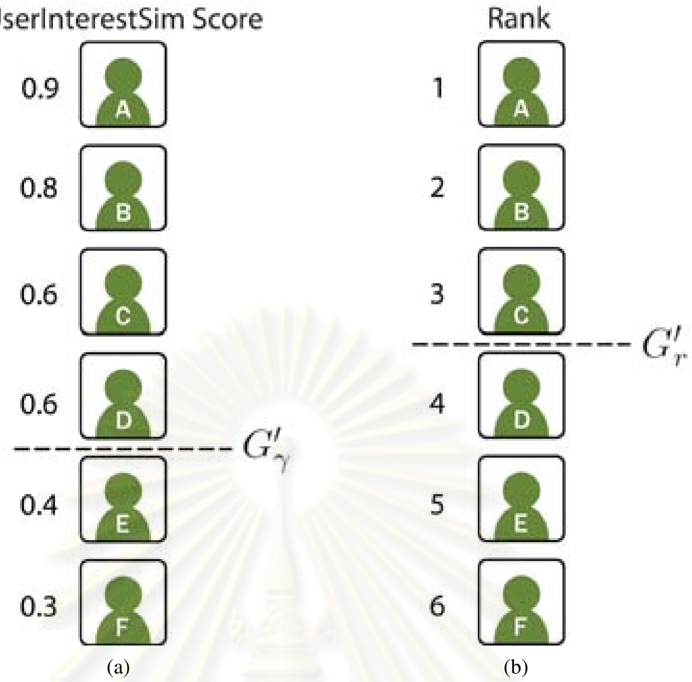


Figure 3.6: Scheme for user removing: (a) cutoff using UserInterestSim threshold, (b) cutoff using rank threshold

similarity between two users in G'_c is defined as equation 3.5.

$$UserSim_{G'_c}(u,f) = \begin{cases} \frac{|S'_u \cap S'_f|+1}{|S'_u|} & \text{if } u \text{ subscribed } f \\ \frac{|S'_u \cap S'_f|}{|S'_u|} & \text{otherwise} \end{cases} \tag{3.5}$$

Where S'_u is the list of u's followees in G'_c and S'_f is the list of the users who are subscribed by author f in G'_c.

We consider the distance as another factor which indicates the possibility that these two users may share the same interest. Thus, we defined the interest factor metric as equation 3.6.

$$InterestFactor(f) = \frac{1}{min\ Distance(u,f)} \tag{3.6}$$

Where $min\ Distance(u,f)$ is the minimum number of edges between u and f.

From all metrics we defined above, the impact of the author f is defined as equation 3.7.

$$ImpactScore(f, G'_c) = InterestFactor(f) \cdot \frac{2UserSim_{G'_c}(u,f)NS_{G'_c}(f)}{UserSim_{G'_c}(u,f) + NS_{G'_c}(f)} \tag{3.7}$$

3.6.1.4 Classification Integrated Score

Combining all the features together, we can determine message relevance according to a particular classifier from classification integrated score (CIS), which is defined as equation 3.8.

$$CIS(m, l, c, f, p_f, G'_c) = \frac{1}{\omega_{c1} + \omega_{c2} + \omega_{c3}} \cdot \begin{bmatrix} \omega_{c_1} \\ \omega_{c_2} \\ \omega_{c_3} \end{bmatrix} \cdot \begin{bmatrix} ContentSim(m, c) + LinkSim(l, c) \\ UserInterestSim(p_f, c) \\ ImpactScore(f, G'_c) \end{bmatrix} \tag{3.8}$$

Where ω_{c_1}, ω_{c_2} and ω_{c_3} are the predefine weight constants used for classification. They controls the weight of content feature, user feature, and community feature respectively. The classification is strict to the content when ω_{c_1} is set to the highest. By setting ω_{c_2} and ω_{c_3} higher, the classification will be less strict for the author whose profile and impact are good enough. This makes the classification less prone to noise, but also better at discovering more messages.

The messages m will be classified as a member of category corresponding to the classifier c, if $CIS(m, l, c, f, p_f, G'_c) \geq \phi_a$, where ϕ_a is the predefined acceptance threshold. The classified messages will be stored in Classified Messages Repository and will be indexed for later use.

3.6.2 Classifier Expansion

In our research, SWEBOK is used to construct the classifiers. However, the content of this document is written in a broad technology-independent manner which may not sufficient enough to classify Microblogging messages which, on the other hand, are written in narrow manner. To overcome this limit, we apply the classifier expansion method which collects the narrow terms from collected messages and uses them to extend the classifier's capability. The conceptual model of classifier expansion process is shown in Figure 3.7 The main idea of this process is to store new terms in Term Cache repository. When a term is important enough, it is moved to Term Extension reposition and calculate its term weight. move it

Figure 3.8 depicts the classifier expansion process. Firstly, the message is tokenized and the Stoplists terms are removed. The terms are checked with the classifier and Term Extension repository. If the classifier already contained the term, it would be discarded. If the term already existed in Term Extension repository, the weight of the term in Term Extension will be adjusted. The left terms are stored in Term Cache repository with the scores corresponding to each classifier.

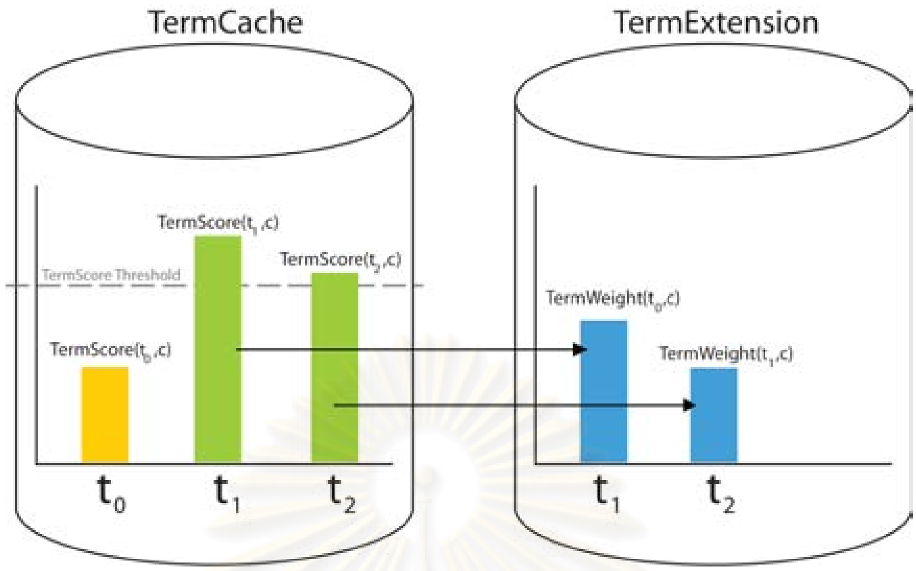


Figure 3.7: Conceptual model of classifier expansion process.

The score of the term in Term Cache repository is calculated from the cumulative impact score of the authors who use that term in their published messages. This score is called term score and is defined as equation 3.9.

$$TermScore(t, c) = \sum_{i=1}^{n} tf(f_i, c) \cdot ImpactScore(f_i, G'_c) \tag{3.9}$$

Where t is the term that is posted by user f_i. $tf(f_i, c)$ is the frequency of term t appearing in the messages that is posted by f_i and is classified by c. Terms in Term Cache will have their scores updated until they exceed the term score threshold ϕ_t. When the score of a particular term t under the classifier c exceeds this threshold, it is moved to Term Extension repository.

When term is moved to Term Extension Repository, its weight must be recalculated. Here, we use the tf-idf for term weighting, however, with a slightly adjustment. The idf is computed based on the number of the document in each classifier c instead of using same idf value for all term in every category. The adjusted weight computation is shown in equation 3.10 and equation 3.11.

$$TermWeight(t, c) = tf(t, c) \cdot idf(t) \tag{3.10}$$

$$idf_c(t) = log_2(n) - log_2(docfreq_c(t)) + 1 \tag{3.11}$$

Where $tf(t, c)$ is the number of occurence of term t in all messages classified by c.

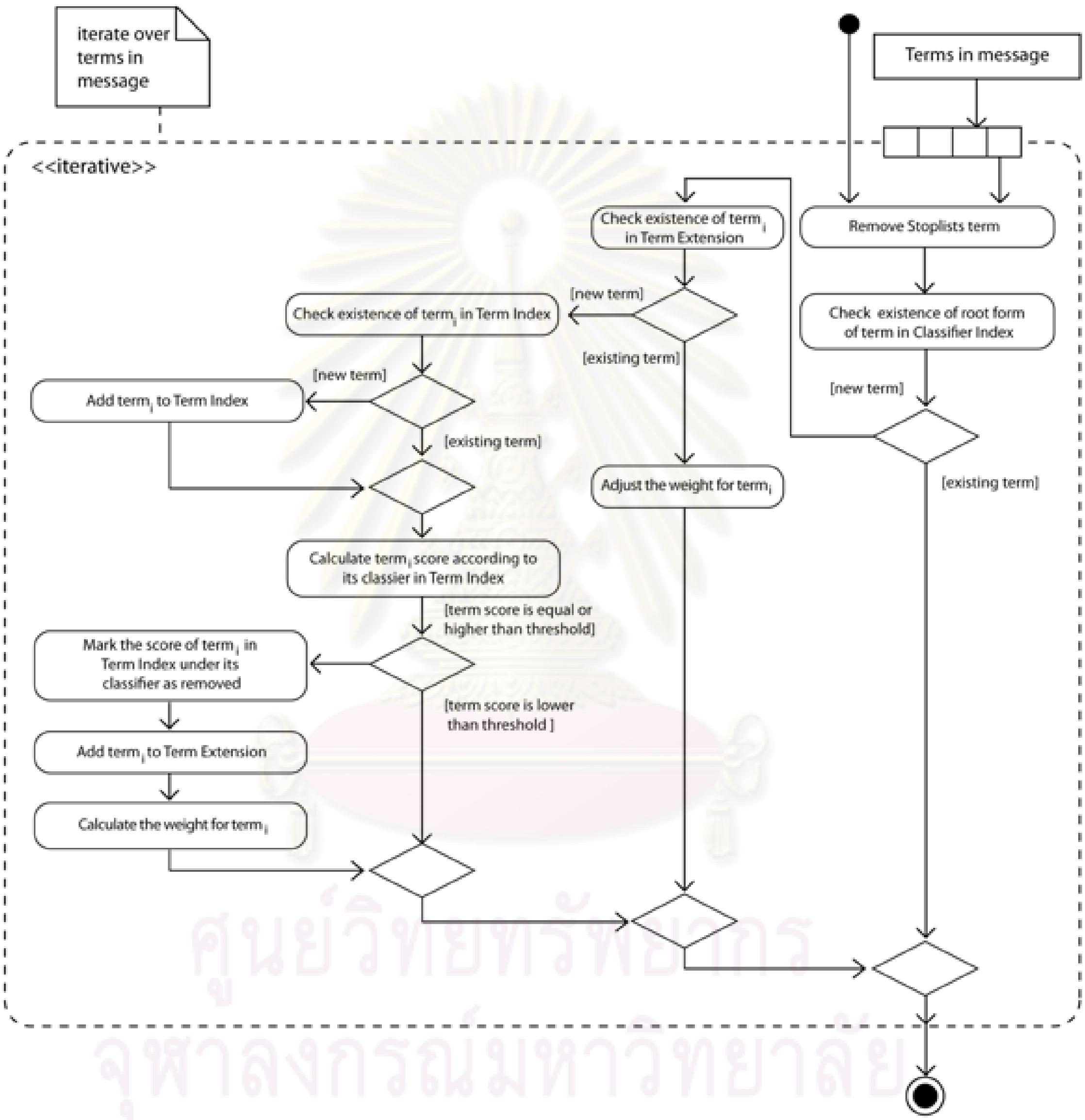


Figure 3.8: Activity diagram of classifier expansion process.

3.7 Social Context Approach : Retrieval Phase

After messages are classified and stored, the information seeker may want to search them. The traditional searching approach that searches the messages according to its textual similarity, as pointed at the beginning of this chapter, may not be sufficient. With the limited length of the message, each term in message often share an identical number of occurrence, i.e., each term often occurs only once or twice. In addition, message on Microblogging can be either an useful opinion or just a story-telling. Although it contains the keyword that matches to user query, it may not really match the user need.

The retrieval process in our work is not different from the IR traditional retrieval except that the similarity score is replaced by the retrieval integrated score (RIS) between the query and messages as depicted in Figure 3.9. The retrieval integrated score is defined as equation 3.12.

$$RIS(m, l, q, f, p_f, G'_q) = \frac{1}{\omega_{s_1} + \omega_{s_2} + \omega_{s_3}} \cdot \begin{bmatrix} \omega_{s_1} \\ \omega_{s_2} \\ \omega_{s_3} \end{bmatrix} \cdot \begin{bmatrix} ContentSim(m, q) + LinkSim(l, q) \\ UserInterestSim(p_f, q) \\ UserSim_{G'_c}(u, f) \end{bmatrix} \tag{3.12}$$

Where ω_{s_1}, ω_{s_2} and ω_{s_3} are the predefine weight constants of content feature, user feature, and community feature respectively. However, in RIS, solely $UserSim$ is used instead of full $ImpactScore$ as it is preferable to base the impact only on personal interests.

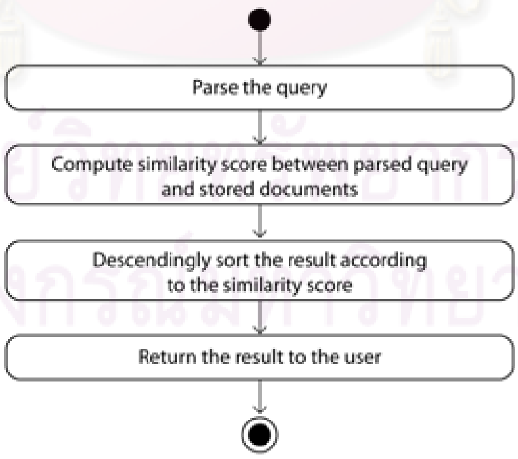


Figure 3.9: Activity diagram of retrieval process.

3.8 Conclusion

In this chapter, we propose the approach for message classification and message retrieval together with new relevance-assessing metrics. In classic IR approach, both message classification and message retrieval can be done by investigating its content feature, i.e., comparing the similarity between a message's content and a classifier's content. Our approach proposes the use of social context which consists of user feature and community feature. The message classification can be, instead, done by investigating the classification integrated score. In the same way, our approach to message retrieval can be done by investigating the retrieval integrated score. In addition, whenever a message is classified, it is parsed to the classifier expansion process which monitors the importance of each new terms. If a new term is important enough, it is added to a classifier so that the classification capability can be increased. Table 3.2 concludes the differences between our approach and classic IR approach. The list of metrics proposed in this work is also shown in Table 3.3.

Table 3.2: The differences between classic IR approach and our approach.

Classic IR Approach	Our Approach
1. The message classification is solely done based on a message's content feature.	1.The message classification is done based on a message's content feature and social context (user feature and community feature).
2. The classifier is static.	2. The classifier is extended by the classifier expansion process.
3. The message is retrieved according to its content feature.	3.The message is retrieved according to its content feature and social context.

Table 3.3: List of metrics proposed in this research.

Traditional Metrics	Description
ContentSim	Assess the similarity between a message's content and a classifier's content.
Proposed Metrics	Description
LinkSim	Assess the similarity between a content of external link that is specified in a message and the content of classifier.
UserInterestSim	Assess the similarity between a content of author's profile and a classifier's content.
ImpactScore	Assess the overall impact of an author according to a topic of interest.
InterestFactor	Assess the possibility that an author and the information seeker will share common interest.
NS	Assess the impact of an author toward other friends in user network according to a topic of interest.
UserSim	Assess the similarity between an author and the information seeker from their subscription behaviors.

CHAPTER IV

EXPERIMENTS

In this chapter, the experiments we conducted to prove our hypotheses are described. Figure 4.1 depicts the process of experiment. Firstly, we begin with the experiment planning which covers objective, design, hypotheses, and metrics. Secondly, the data preparation process for the experiment is described. Lastly, as the experiment in our research is divided into two parts, the first part, the classification evaluation experiment is described followed with the retrieval evaluation experiment. Each section of these experiments covers the procedure, the control factors, the experimental results, the experimental results analysis, the experimental result summary, and is ended up with the discussion.

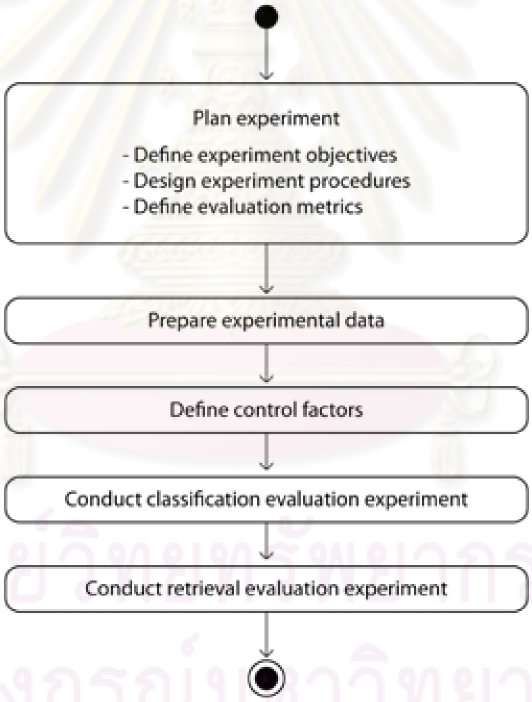


Figure 4.1: Activity diagram of experiment process

4.1 Experiment Planning

4.1.1 Objectives

The objective of the experiments are to evaluate the proposed framework for its classification and retrieval effectiveness compared to the traditional IR approach and to assess if the improvement was statistical significant.

4.1.2 Design

The experiment is divided into two parts: the classification evaluation and the retrieval evaluation. Each of them is described as follows.

4.1.2.1 Classification Evaluation

There are three objectives we want to assess in this part. Firstly, to assess whether the use of social context without classifier expansion gives a better effectiveness than the baseline. Secondly, to assess whether the use of social context and classifier expansion gives a better effectiveness than the baseline. Thirdly, to assess whether the classifier expansion gives a significant difference compared to solely use of social context. Thus, we define three classification treatments as follows.

1. **Baseline treatment** (CT_0). The classification under this treatment is done solely by textual similarity comparison (content feature).
2. **Social context treatment** (CT_1). The classification under this treatment is done using the classification integrated score as described in section 3.6.1.4.
3. **Social context with classifier expansion treatment** (CT_2) . The classification under this treatment is done using the classification integrated score with the classifier expansion process applied as described in section 3.6.2.

The classification evaluation will be conducted in the following ways.

1. The data for evaluation are collected.
2. The collected data is evaluated by the expert for their relevances according to the categories in SWEBOK.
3. The data are parsed to each classification treatment and the results are recorded.
4. The classification results of each treatment are compared to those done by the expert for their effectiveness. After that, the classification effectiveness of each treatment is compared toward each other.

4.1.2.2 Retrieval Evaluation

The main purpose of retrieval evaluation is to determine whether the proposed retrieval model give a better retrieval effectiveness compared to the traditional IR model. Thus, we define two retrieval treatments as follows.

1. **Baseline treatment** (RT_0). The retrieval under this treatment is done solely by textual similarity comparison (content feature).
2. **Social context treatment** (RT_1). The retrieval under this treatment is done using the retrieval integrated score as described in section 3.7.

The retrieval evaluation will be conducted in the following ways.

1. The classification evaluation is done and the classified messages are stored in the repository.
2. The queries are generated from collected messages.
3. The queries are submitted for to each retrieval treatment.
4. The result according to the query is shown. The expert manually judges the relevance of each retrieved document.
5. The retrieval effectiveness between each treatment is compared.

4.1.3 Metrics

4.1.3.1 Classification Evaluation Metrics

The effectiveness of message classification is judged from its correctness compared to the evaluated classification done by expert. The classification is correct if the classification by the treatment is exactly the same as by the expert. Therefore, the correctness can be defined in two perspectives as follows.

1. **True positive correctness (TP)**. For a given message m and category c, the classification of treatment CT_i is true positive if both treatment CT_i and expert classify message m as a member of category c.
2. **False negative correctness (FN)**. For a given message m and category c, the classification of treatment CT_i is false negative if both treatment CT_i and expert classify message m as not a member of category c.

Given a treatment CT_i and a category c, we define three metrics for classification evaluation.

1. **Precision.** The precision of treatment CT_i for category c is a ratio between the number of true positive correctness and the total number of messages classified as a member of category c by the expert. The precision is defined as the following equation. 4.1.

$$precision_{CT_i}(c) = \frac{\text{total number of true positive items by } CT_i}{\text{number of items classified as a member of } c \text{ by expert}} \tag{4.1}$$

2. **Fallout.** The fallout of treatment CT_i for category c is a ratio between the number of false negative correctness and the total number of messages classified as not a member of category c by the expert. The fallout is defined as equation 4.2.

$$fallout_{CT_i}(c) = \frac{\text{total number of false negative items by } CT_i}{\text{number of items classified as not a member of } c \text{ by expert}} \tag{4.2}$$

3. **Harmonic mean.** The harmonic mean of treatment CT_i for category c reflects the overall effectiveness in both true positive and false negative perspectives. Harmonic mean is defined

as the following equation. 4.3.

$$F_{CT_i,\beta}(c) = \frac{1}{\beta.\frac{1}{precision_{CT_i}(c)} + (1-\beta).\frac{1}{fallout_{CT_i}(c)}} \tag{4.3}$$

Where β is the weight constant which its value is between 0 and 1. In this research, we weight precision and fallout equally. Thus, β is fixed to 0.5.

4.1.3.2 Retrieval Evaluation Metrics

Mentioned in section 2.1.3, the r-precision metrics is suitable according to the retrieval evaluation objective. However, r-precision has one drawback that the rank of the item in result set is discarded. The r-precision of two treatments are equal if their results share identical number of relevance item. Thus, we define weighted r-precision (WPR) that considers the rank of the item in calculation. It can be computed as equation 4.4.

$$WPR_{RT_i}@r = \frac{1}{r} \cdot \sum_{j=1}^{r} (r+1-j) \cdot (relevance(j)) \tag{4.4}$$

Where r is a document cutoff value and $relevance(j)$ is the relevance of the document in j position of the result set. $relevance(j)$ equals to 1 if the retrieved document at position j is relevant to query q and equals to 0 if the retrieved document at position j is not relevant.

With weighted r-precision, the value goes high when relevance documents float at top of result set. The value goes low when relevance documents fall down to bottom of the result set. The penalty of the rank is in linear regression.

Another metric with the same idea as WPR is discounted cumulative gain (DGC). The difference between them is that DGC has its rank penalty as logarithmic reduction. DGC is computed as equation 4.5.

$$DCG_{RT_i}@r = \sum_{j=1}^{r} \frac{2^{relevance(j)} - 1}{log_2(j+1)} \tag{4.5}$$

In our experiment, the retrieval effectiveness is judged with these two metrics with the document cutoff value $r \in \{5, 10, 20\}$.

4.2 Data Preparation

4.2.1 Data Preparation for Classification Evaluation

In order to evaluate the classification effectiveness, we prepared the data set which consists of the messages, the users and their relations from Twitter. The preparation process is depicted by Figure 4.2. Firstly, the dummy user u is created which we assume that this user is the information seeker who use the system. Secondly, titles of each knowledge area are submitted as queries on Twitter search. Authors of messages in the search result are scrutinized and selected when they meet selection criteria. Thirdly, we subscribe u to every selected user. Next, information of every user such as full name, screen name, and subscription list is retrieved. Finally, recent 3,000 messages of each user are fetched from Twitter. The crawler which periodically crawls information via Twitter API were created. We use it to collect the information from March to April 2010.

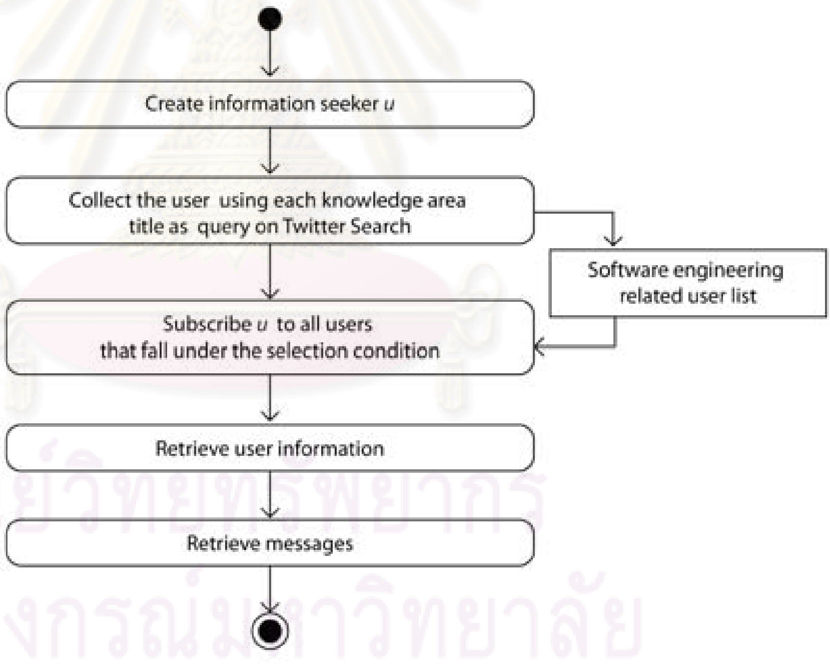


Figure 4.2: Activity diagram of data preparation process.

4.2.1.1 Users

User selection is crucial. As software engineering domain is focused, the selected user must be related to software development. To achieve this, ten SWEBOK's knowledge area titles are submitted to Twitter search together with some narrow terms such as ‘CMMI’, ‘TDD’, and ‘agile’. After result is returned from the search, each author is scrutinized. There are two criteria to decide whether an author should be selected.

1. The author must be related to software development. This can be decided by investigating user profile and recent messages.
2. The author must have at least two subscriptions (follow or followed) to the previously selected users.

With these criteria, totally 141 users with 528 subscription relations (excluding the subscription from the created dummy user) are collected. Full list of user is shown in Appendix A.

4.2.1.2 Messages

After list of users is acquired, their recent messages are collected. Due to the API limitation, the maximum number of messages that can be retrieved is 3,000 messages per use. Total number of messages that could be collected is 208,167 messages (1,476 messages per user by average).

From the collected messages, we divided them into two groups. The first group, denoted as M_e, consists of the most recent 100 messages from all users. The second group, denoted as M_p, consists of the messages that do not fall into the first group. Totally, there are 12,842 messages in M_e and 190,295 messages in M_p.

The relevance of messages in M_e are classified according to each knowledge area in SWEBOK by the expert. A message can be classified as a member of multiple categories. The number of message evaluated under each category is shown in Table 4.1. Examples of messages in each category are included in Appendix B.

The uses of M_p and M_e are shown in Figure 4.3. Messages in M_p is used for profile construction as described in section 3.5. After profiles of all user are constructed, messages in M_e are sequentially parsed to classification process ordered by their created dates.

Table 4.1: The number of evaluated message in each category.

Category	Number of messages
Software Requirement	68
Software Design	1,022
Software Construction	2,412
Software Testing	390
Software Maintenance	199
Software Configuration Management	119
Software Engineering Management	182
Software Engineering Process	92
Software Engineering Tools and Methods	1,118
Software Quality	260
Total	5,862

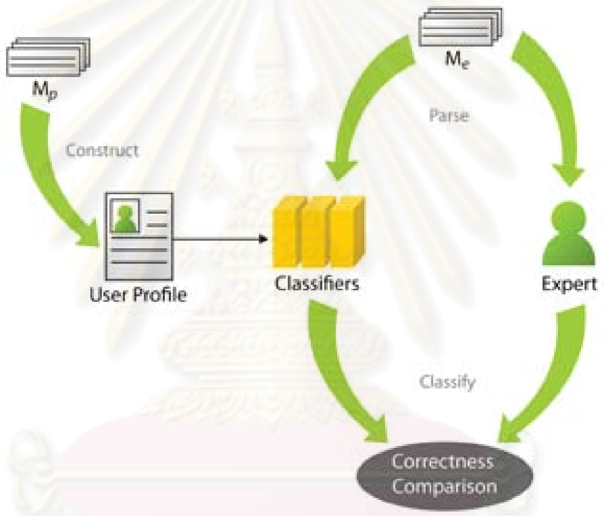


Figure 4.3: Usages of M_e and M_p.

4.2.2 Data Preparation for Retrieval Evaluation

The data that are needed to be prepared for retrieval evaluation are the messages for being queried and the queries.

4.2.2.1 Messages

The messages in M_e are also used for retrieval evaluation.

4.2.2.2 Queries

For queries, the query preparation is done as shown in Figure 4.4. Firstly, the terms of messages in M_p are extracted then they are sorted in descending order by frequency. After that, 50 query terms are manually selected. We use the following criteria for query selection.

1. The query must be monogram (a sequence of characters without white space in-between).
2. The query must be a noun.

Occurrence frequency of the selected queries varies between 123 to 576 times. Both broad terms and narrow terms are selected. We use only monogram query as we want to remove the effect of term context that helps making the query less ambiguous. We expect that the use of social context may help reducing the term ambiguity as the context is compared based on the user interest. Full list of query is shown in Appendix C.

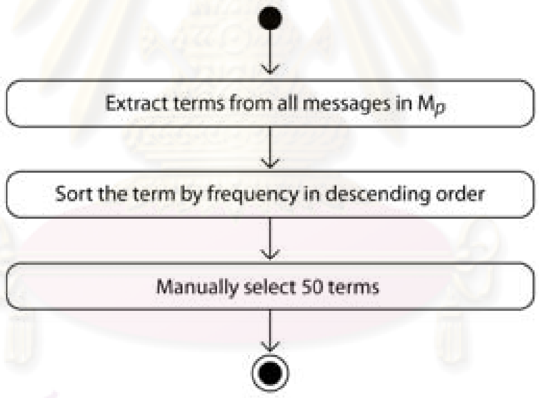


Figure 4.4: Activity diagram of query preparation process.

4.3 Classification Evaluation

The procedure of classification evaluation is shown in Figure 4.5. Firstly, after data are prepared, the expert evaluates all messages in M_e. We denoted $M(C)$ as the set of message that is evaluated as a member of one or more categories $c \in C$. Simultaneously, all messages in M_p are used for profile construction as described in section 3.5. Next, all messages in M_e, sorted in ascending order by created date, are classified by each treatment. Then, the result of classification from every treatments are compared to those done by the expert for the effectiveness. Finally, effectiveness of each treatment is compared.

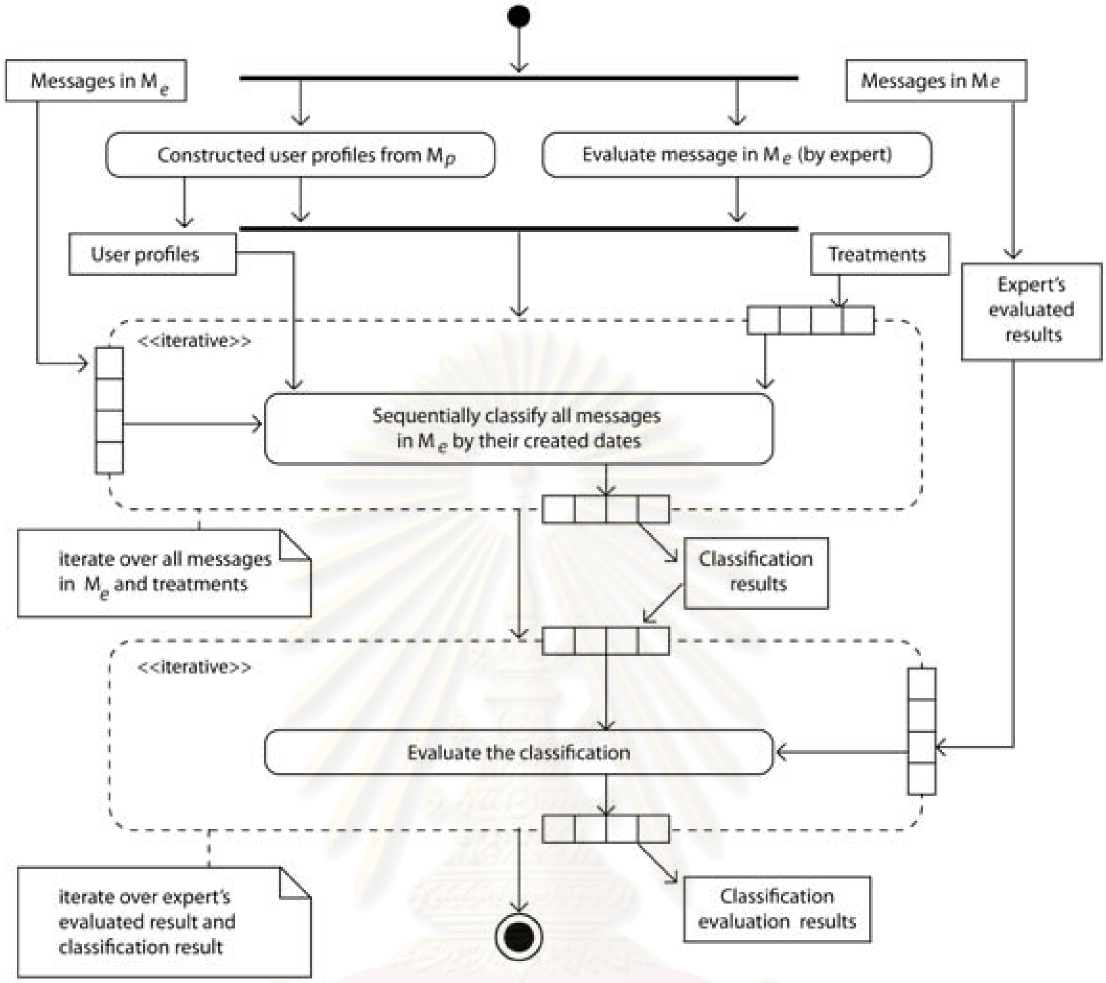


Figure 4.5: Activity diagram of classification evaluation procedure.

4.3.1 Environment

There are many control factors that are needed to be set before experiment takes place. Table 4.2 summarizes all the control factors and their values. The categories used in classification are the knowledge areas in SWEBOK that are used in classifier generation. There are three treatments in this experiment which are baseline treatment, CT_0, social context treatment, CT_1, and social context treatment with classifier expansion, CT_3. The first treatment, CT_0, is fixed with the weight set $[\omega_{c_1}, \omega_{c_2}, \omega_{c_3}] = [1, 0, 0]$ (using only content feature), while others treatment is assigned with weight set $[\omega_{c_1}, \omega_{c_2}, \omega_{c_3}] = [1, 1, 1]$ as we want to assess the effect of all features when they are used equally. We decide to use cutoff position for user network reduction as described in section 3.6.1.3 where the value of G_r is fixed as 100. The acceptance threshold ϕ_a is differently selected for each treatment. For CT_0, ϕ_{a_0} is set as 0.03. For CT_1, ϕ_{a_1} is set as

Table 4.2: Control factors for classification evaluation.

Control Factor	Description	Value
$C = \{c_0, c_1, ...c_9\}$	The set of category of the message that are used in message classification.	c_0 = 'Software Requirement', c_1 = 'Software Design', c_2 = 'Software Construction', c_3 = 'Software Testing', c_4 = 'Software Maintenance', c_5 = 'Software Configuration Management', c_6 = 'Software Engineering Management', c_7 = 'Software Engineering Process', c_8 = 'Software Engineering Tools and Methods', c_9 = 'Software Quality'
M_e	The messages used for evaluation.	$\| M_e \|= 12,842$
M_p	The messages used for profile construction.	$\| M_p \|= 190,295$
G_r	Cutoff position used in user network reduction.	100
ϕ_{a_0}	Acceptance threshold used for baseline treatment.	0.03
ϕ_{a_1}	Acceptance threshold used for social context treatment.	0.065
ϕ_{a_2}	Acceptance threshold used for social context treatment with classifier expansion.	0.1
ϕ_t	Term score threshold used in classifier expansion process.	1.5
MPR	The number of message to be parsed before the classifier is refreshed.	400

0.065. These two values are selected from the mean of integrated score of all messages classified by the expert. CT_0 and CT_1 can not use the same value of ϕ_a. If ϕ_{a_0} is set to 0.065, the number of message classified by CT_0 will be too low as scores of most message from CT_0 are low. On the other hand, if ϕ_{a_1} is set to 0.03, the number of message classified by CT_1 will be too high as scores of most message from CT_1 are high. However, for CT_2, the mean value of integrated score can not be calculated because the integrated score is depended on ϕ_{a_2} value. Therefore, we decide to set CT_2 equally to CT_1 as both of them use same weight set. Therefore, the mean of integrated score of CT_1 and CT_2 should not be much different. For classifier expansion process, there are two factors to consider. Firstly, the term score threshold ϕ_t of CT_2 is set to 1.5. We decide to allow term that occurs around five times to be added as extended term. The number 1.5 is calculated by multiplying 5 with 0.3 which is the average impact score of messages in $M(C)$. Secondly, the message per refresh MPR is set to 400. This means that weights of all term in each classifier will be recalculated every times 400 messages are parsed. More detail about MPR is included in section 4.3.6.

4.3.2 Experimental Tool

To support the classification evaluation experiment, the command line tool is for message classification is created. This tool is implemented with Java and Apache Lucene. Its architecture is depicted by Figure 4.6. Messages in M_e and the evaluated results done by the expert are stored in the file system which is done by File System layer. Lucene layer is the interface layer that provides the access to the stored data. Data Model layer is the wrapper layer that maps the stored data to objects and Message Classification layer classifies the message according to the configured parameters. As the classification evaluation can be run in batch mode, the only parameters required for the tool are the weights $[\omega_{c_1}, \omega_{c_2}, \omega_{c_3}]$, ϕ_a and ϕ_t. The usage of the tool with its input and output is depicted by Figure 4.7. When the experiment is performed, each message in M_e is parsed and classified with these parameters, then the classification result is compared to the evaluation record in $M(C)$. Figure 4.8 shows the screenshot of the tool's code where the parameters of each treatment can be configured and run. After the tool performs its task, it returns the output in comma separated format (.csv) as shown in Figure 4.9.

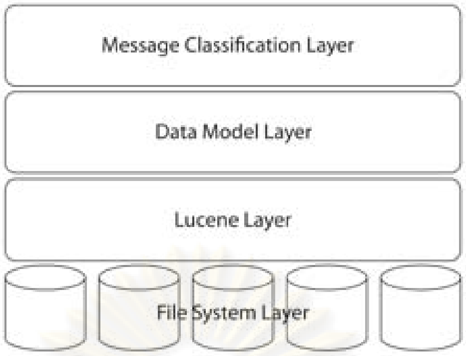


Figure 4.6: Classification evaluation tool architecture.

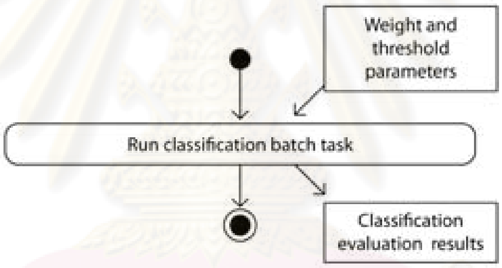
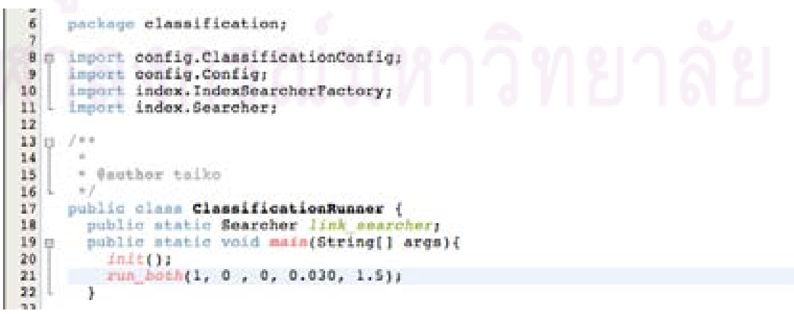
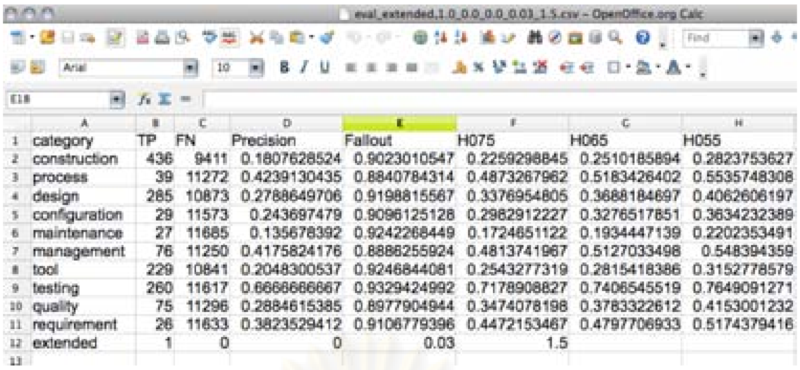


Figure 4.7: Classification evaluation tool usage with input and output.



```
6   package classification;
7
8   import config.ClassificationConfig;
9   import config.Config;
10  import index.IndexSearcherFactory;
11  import index.Searcher;
12
13  /**
14   *
15   * @author taiko
16   */
17  public class ClassificationRunner {
18    public static Searcher link_searcher;
19    public static void main(String[] args){
20      init();
21      run_both(1, 0 , 0, 0.030, 1.5);
22    }
```

Figure 4.8: Screenshot of the tool's code. Parameters for each treatments can be set at line 22.



category	TP	FN	Precision	Fallout	H075	H065	H055
construction	436	9411	0.1807628524	0.9023010547	0.2259298845	0.2510185894	0.2823753627
process	39	11272	0.4239130435	0.8840784314	0.4873267962	0.5183426402	0.5535748308
design	285	10873	0.2788649706	0.9198815567	0.3376954805	0.3688184697	0.4062606197
configuration	29	11573	0.243697479	0.9096125128	0.2982912227	0.3276517851	0.3634232389
maintenance	27	11685	0.135678392	0.9242268449	0.1724851122	0.1934447139	0.2202353491
management	76	11250	0.4175824176	0.8886255924	0.4813741967	0.5127033498	0.548394359
tool	229	10841	0.2048300537	0.9246844081	0.2543277319	0.2815418386	0.3152778579
testing	260	11617	0.6666666667	0.9329424992	0.7178908827	0.7406545519	0.7649091271
quality	75	11296	0.2884615385	0.8977904944	0.3474078198	0.3783322612	0.4153001232
requirement	26	11633	0.3823529412	0.9106779396	0.4472153467	0.4797706933	0.5174379416
extended	1	0	0	0.03	1.5		

Figure 4.9: Output after the tool is run.

4.3.3 Experimental Result

Figure 4.10 shows the precision comparison among treatments. CT_2 gives the highest precision in most categories except in Software Testing, Software Configuration Management, and Software Management. CT_1 gives the lower score compared to CT_2, yet its precision is still higher than CT_0 except in Software Configuration Management and Software Quality category. It also gives the highest precision in Software Testing. Even CT_0 gives low score, it still gives the best precision for Software Configuration Management category. The average precision of CT_0, CT_1, and CT_2 are 0.31, 0.39 and 0.41 respectively.

Although both CT_1 and CT_2 result in higher precision than CT_0, they must trade their fallout off. Figure 4.11 shows the fallout comparison among treatments. CT_0 gives the highest fallout which its average equals to 0.91. Fallout of CT_1 and CT_2 drop to 0.81 and 0.78 respectively by average. The decreasing of fallout in these treatments is the result of the increment of message score. Not only the correct message that gets its score increased from social context, but also the incorrect message that has good social context. This increment makes them exceed the threshold and pass the classification.

The harmonic mean sums both precision and fallout together to get an overall effectiveness. It indicates, as depicted by Figure 4.12, that CT_2 is the best in six of all categories which are Software Requirement, Software Design, Software Construction, Software Maintenance, Software Engineering Tools and Methods and Software Quality. CT_0 hits the highest harmonic mean in Software Testing, Software Configuration and Software Quality while CT_1 achieves the highest in Software Engineering Management and Software Engineering Process category. Full classification scores of each treatment are included in Appendix D.1.

As a result from CT_2, new terms are added to the classifiers. All list of top 50 new terms for each classifier, together with top 50 terms from classifier itself and top 50 terms of the messages that are a member of the corresponding category, are shown in Appendix D.2.

4.3.4 Experimental Result Analysis

From the results reported in the previous section, we use statistical analysis to confirm three hypotheses as follows.

1. Social context treatment, CT_1, has better classification effectiveness than baseline treatment, CT_0. From this hypothesis, we define null and alternative hypothesis as

$$\begin{aligned} H_0 &: \mu_0 \geq \mu_1 \\ H_1 &: \mu_0 < \mu_1 \end{aligned} \tag{4.6}$$

where μ_0 is the mean of harmonic mean of CT_0 and μ_1 is the mean of harmonic mean of CT_1.

2. Social context with classifier expansion treatment, CT_2, has better classification effectiveness than baseline treatment, CT_0. From this hypothesis, we define null and alternative hypothesis as

$$\begin{aligned} H_0 &: \mu_0 \geq \mu_2 \\ H_1 &: \mu_0 < \mu_2 \end{aligned} \tag{4.7}$$

where μ_0 is the mean of harmonic mean of CT_0 and μ_2 is the mean of harmonic mean of CT_2.

3. Social context with classifier expansion treatment, CT_2, has better classification effectiveness than the social context treatment, CT_1. From this hypothesis, we define null and alternative hypothesis as

$$\begin{aligned} H_0 &: \mu_1 \geq \mu_2 \\ H_1 &: \mu_1 < \mu_2 \end{aligned} \tag{4.8}$$

where μ_1 is the mean of harmonic mean of CT_1 and μ_2 is the mean of harmonic mean of CT_2.

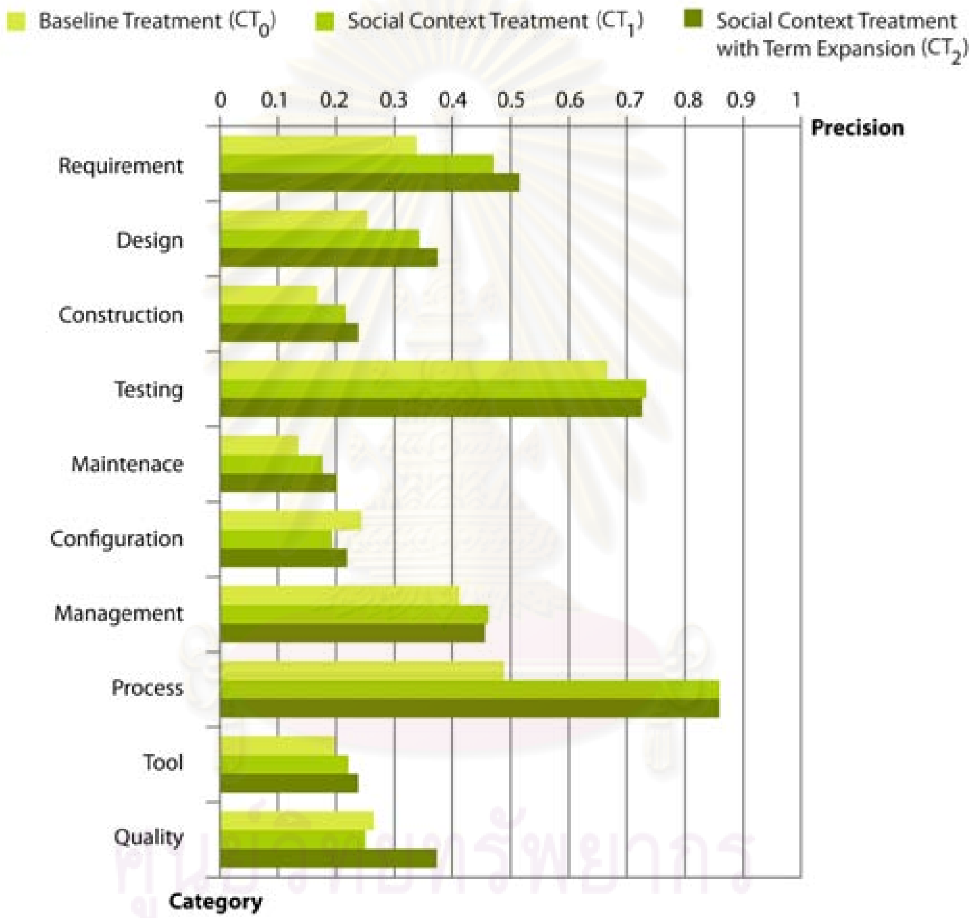


Figure 4.10: Precision comparison among treatments.

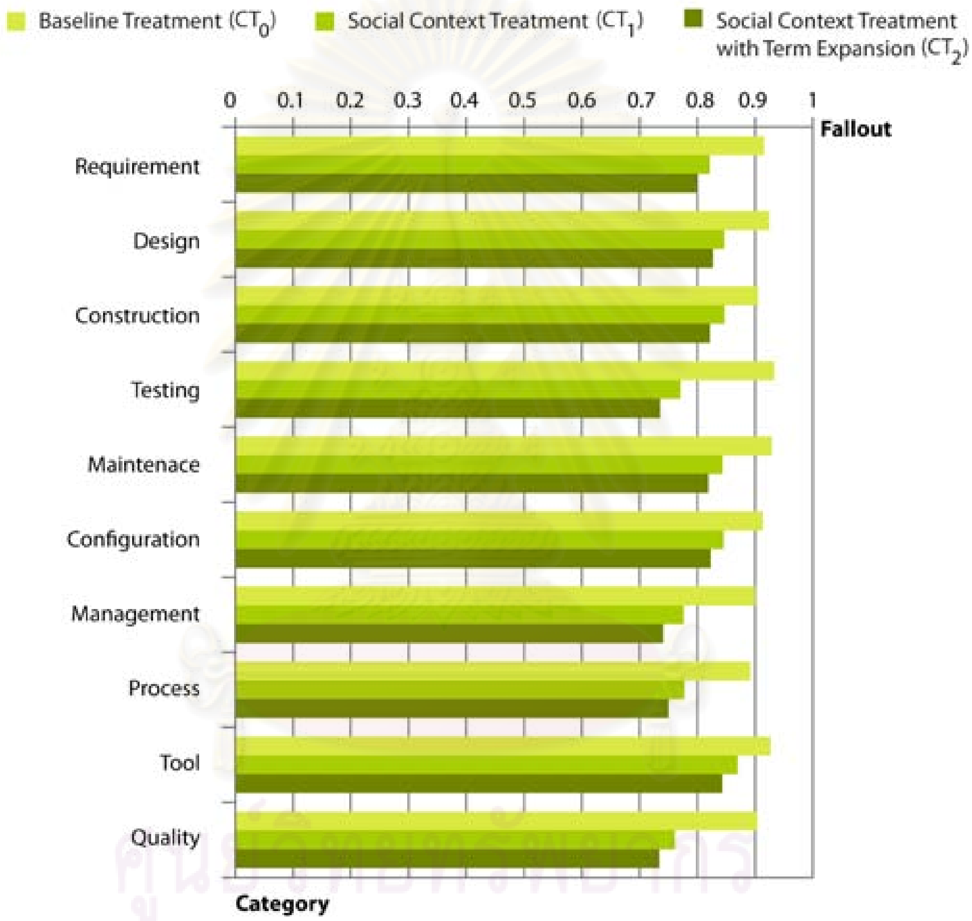


Figure 4.11: Fallout comparison among treatments.

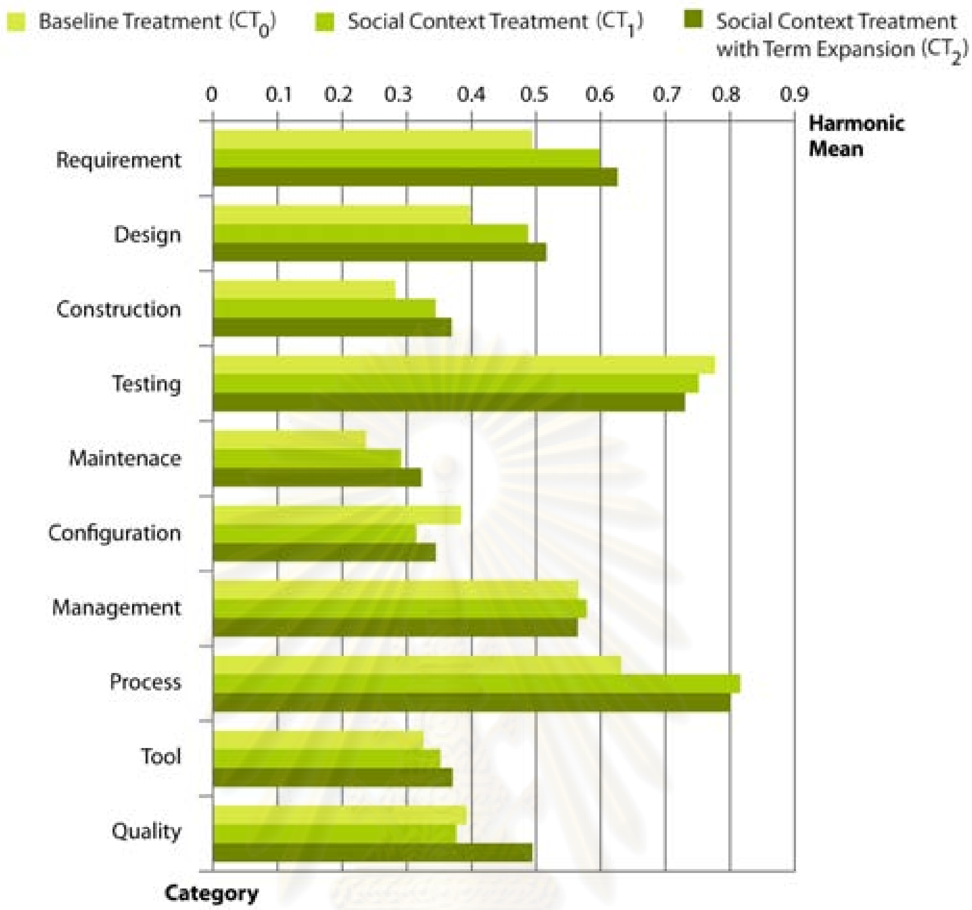


Figure 4.12: Harmonic mean comparison among treatments.

Paired t-test is selected for this hypothesis testing as the test is conducted with the same dataset for each treatment. However, as the number of sample unit is small (10 categories), the normality of data can not be assumed. Therefore, we need to check normality as paired t-test required that the populations must follow normal distribution.

To confirm this assumption, the null hypothesis that the selected population follows normal distribution is set. Saphiro-Wilk test is applied which its result is shown in Table 4.3. The significant level α is set to 0.05. As all p-values are higher than 0.05, null hypotheses are not rejected. Therefore, we can conclude that the harmonic mean values of all treatments are likely to follow the normal distribution.

The result of the paired t-test is show in Table 4.4. All alternative hypotheses are accepted at 0.05 significant level (all p-value are less than α).

Table 4.3: Saphiro-Wilk test result.

Treatment	W	P-value	H_0
CT_0	0.9422	0.5777	Not reject
CT_1	0.8896	0.1679	Not reject
CT_2	0.9203	0.3598	Not reject

Table 4.4: Hypothesis testing result for message classification.

Hypothesis	T	Df	P-value	H_1
CT_1 has better classification effectiveness than CT_0	1.834500	9	0.049890	Accepted
CT_2 has better classification effectiveness than CT_0	2.812700	9	0.010400	Accepted
CT_2 has better classification effectiveness than CT_1	1.833900	9	0.049900	Accepted

4.3.5 Experimental Result Summary

According to the experimental results in section 4.3.3 and the experimental result analysis in section 4.3.4, the result of the experiments can be summarized as follows.

1. It is statistically confirmed that both social context treatment and social context with classifier expansion treatment have better classification effectiveness than baseline treatment. Therefore, by using solely social context or social context with classifier expansion for message classification, the classification effectiveness is increased.
2. It is statistically confirmed that social context with classifier expansion treatment results in higher classification effectiveness than social context treatment. Thus, classifier expansion can help improving the classification effectiveness.
3. Both social context treatment and social context with classifier expansion treatment results in higher precision than baseline in most category. However, the fallout is traded off with the capability to classify more messages.

4.3.6 Discussion

According to the results from the classification effectiveness evaluation experiment, there are some interesting points for discussion as follows.

4.3.6.1 Effect of Social Context

Message classification can be viewed as a clustering problem. The preliminary idea behind the proposal of social context is that by invoking social context, the distance between each messages will be changed, i.e, the score distribution of the message will be broader.

The message should get higher score when one or both of these conditions are met.

1. The author of the message has the profile that is more similar to the classifier.
2. The author has high impact.

On the other hand, the message should get lower score when one or both of the above conditions fail. To inspect this assumption, we create the box plot of classification integrated score of all messages in $M(C)$ as shown in Figure 4.13. Two lines showing the acceptance threshold values ϕ_{a_0}, ϕ_{a_1} and ϕ_{a_2} ($\phi_{a_1} = \phi_{a_2}$) are also marked in this figure.

The box plot illustrates distributions of classification integrated score of each treatment. Area in each box shows score distribution of half number of all messages. The line inside the box indicates the median of the score. The whiskers, two vertical lines at the beginning and at the end of the horizontal line that the box lies on, indicate the 25^{th} and 75^{th} quartile. The box position indicates the skew of the score. If the box located to the left side of the container line, score distribution skews right. On the other hand, score distribution skews left when the box position locates to the right side of the container line. The dots show the outliers: the scores that are too low or too high which cause the misleading value of mean. For example, considering Software Engineering Management category of CT_0, the box area and position that is near the left whisker indicates that its score distribution skews to the right. The median states that half of the score lies at the beginning of the distribution curve. From this interpretion, we can imply that most of message in this category has low score.

It is shown that after invoking social context, score distributions are changed. The box area of CT_1 and CT_2 grow larger and the locations of the box are moved to the right. This means that some messages get their scores increased by the social context and classifier expansion. The positions of the boxes together with the acceptance threshold lines also enable us to imply how the classification could be. For instance, consider the boxes of Software Requirement category, we can imply that the classification done by CT_0 should found only less than half of all messages as the box area (50 percent of message) is lower than ϕ_{a_0}. This is in compliance with the actual result of CT_0 that only 38 percent of message are found. This box plot also illustrates that CT_1 and CT_2 could classify more message than CT_0.

4.3.6.2 Characteristics of Messages in M(C)

In this research, we evaluate the message classification of the social context treatments that treats each feature equally. However, in practice, the weight of each feature should be set differently according to the characteristics of messages in each information seeker's environment.

We investigate our messages in $M(C)$. Each feature of them are scrutinized as shown in Figure 4.14. For all categories, it is indicated that the scores of content feature locate in the lower position (the box area is near the left whisker) than those of user feature and community feature. Software Testing Category has large area of content feature and user feature score distribution. This means that the score is varied in higher degree than others. Intuitively, the possibility that the distribution of classification integrated score will be broader is low. However, for other categories, the distribution of user feature and community feature are narrow, and are higher than the content feature. Thus, we can expect, with higher possibility, that the classification integrated score of these categories will get broader. This assumption can be seen in Figure 4.13. However, it is not true for Software Configuration Management category. The reason behind this is that there is an author who contributes more number of message in this category than others. This is obviously illustrated by the box of user feature and community feature of Software Engineering Management category. The box area is narrow and is at the median value. We can imply that this author contributes more than 50 percent of message under this category. Thus, when the integrated score is computed, the scores of message from this author group up together while leaving the scores of other messages as outliers.

In addition to the distribution, we compute average score of each feature as shown in Table 4.5. Considering Software Testing category, its average content feature is the highest. As a result, CT_0 can classify them even without the aid of social context. These average score leads to us the decision on how the classification integrated score function should be tuned.

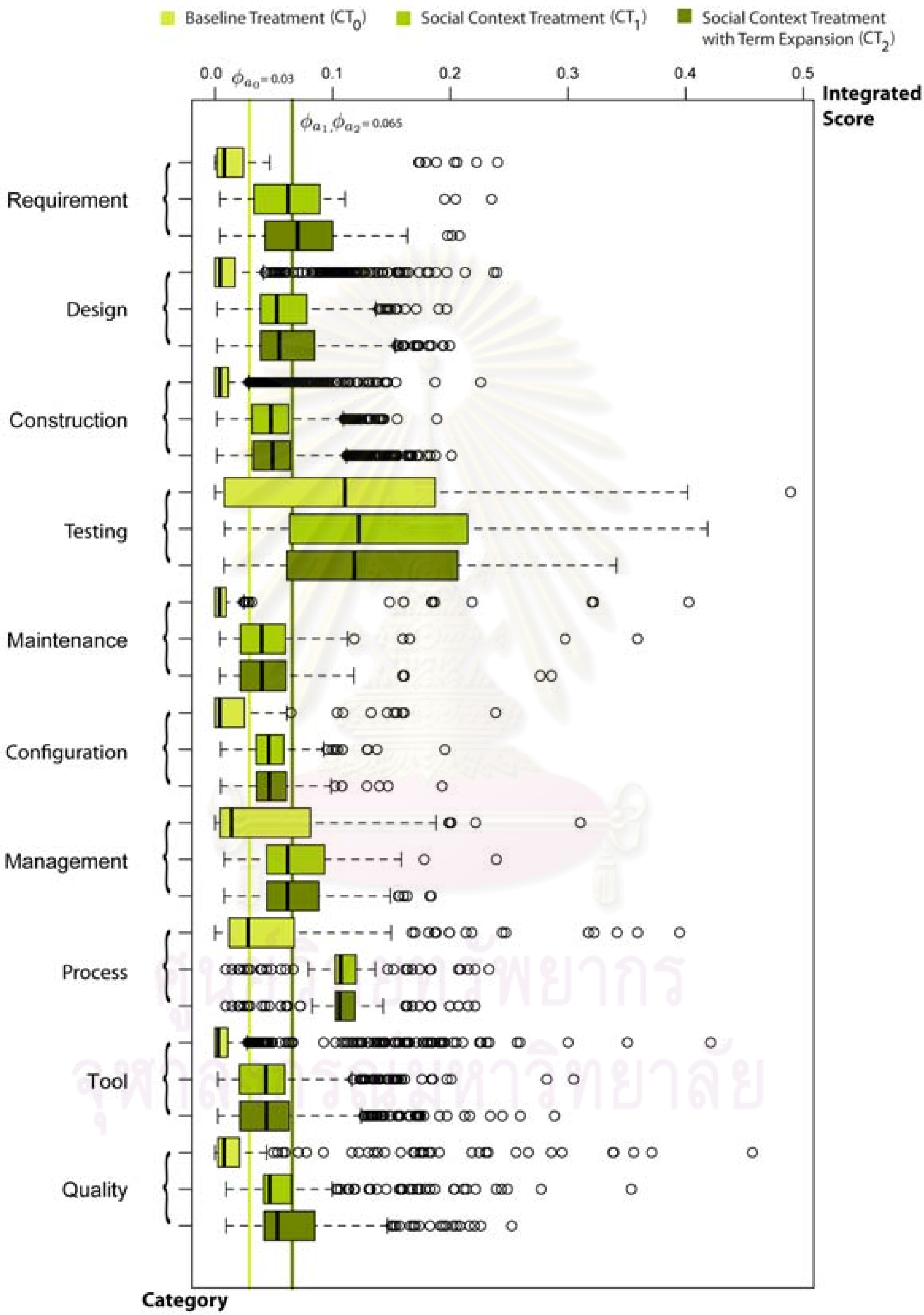


Figure 4.13: Box plot showing the integrated score distribution comparison among treatments.

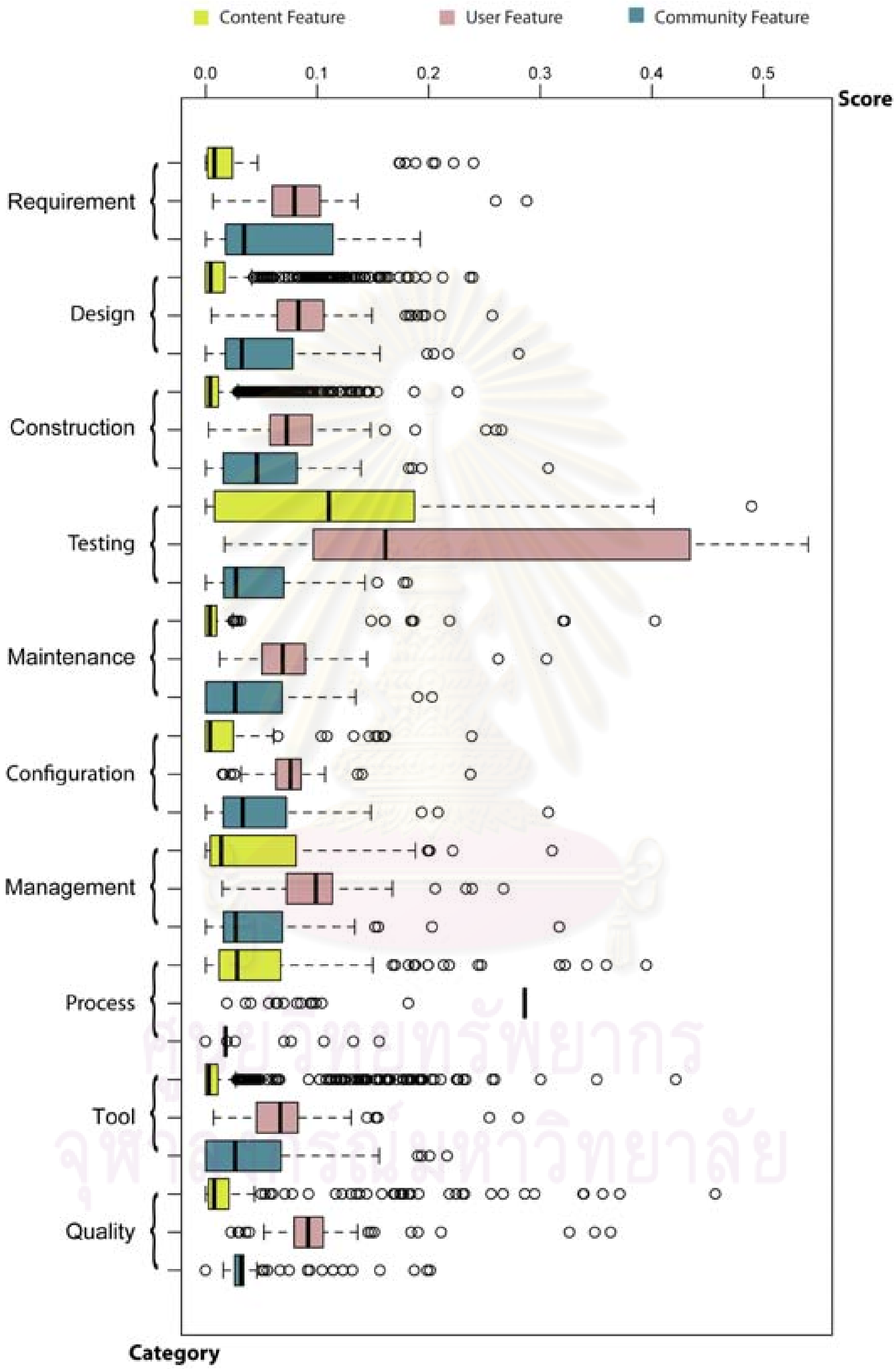


Figure 4.14: Box plot showing the score distribution of content feature, user feature and community feature of message in $M(C)$.

Table 4.5: Average score of each feature.

Category	Content Feature	User Feature	Community Feature
Requirement	0.0332	0.0928	0.0625
Design	0.0192	0.0916	0.0585
Construction	0.0113	0.0794	0.0550
Testing	0.1111	0.2504	0.0419
maintenance	0.0172	0.0747	0.0437
Configuration	0.0216	0.0728	0.0518
Management	0.0436	0.1001	0.0488
Process	0.0688	0.2385	0.0215
Tool	0.0199	0.0782	0.0424
Quality	0.0342	0.1178	0.0378
Average	0.0359	0.1196	0.0463

4.3.6.3 Tuning

According to the Table 4.5, it is shown that, in every category, average score of content feature is the lowest. Given the weight of each feature equally ($[\omega_{c_1}, \omega_{c_2}, \omega_{c_3}]$ = [1,1,1]) in this situation results in the higher value of classification integrated score. Even the result from our experiment shows that the effectiveness is better, such integrated score may be over-tuning. If author has high user feature, or high community feature, or both, message may pass the classification even it is not a member of a particular category. Although user feature and community feature help distinguishing the message, their effects should be limited in the less portion than the effect of content feature. This can be done by giving a proper tuning.

Selecting different weight combination also results in different selection of acceptance threshold value. This threshold should be selected approximately by the mean of the dominant feature. For example, according to Table 4.5, ϕ_{a_0} of baseline treatment (the weight set of baseline treatment is [1,0,0]) is selected as 0.035 as the score of content feature is dominant.

According to this assumption, we conduct another experiment by setting the weight $[\omega_{c_1}, \omega_{c_2}, \omega_{c_3}]$ of CT_1 from [1,1,1] to [10,1,1] in order to raise the effect of content feature up. Figure 4.15, Figure 4.16 and Figure 4.17 shows the precision, fallout and harmonic mean of CT_1 with [10,1,1] configuration compared to those of CT_0 and CT_1 with [1,1,1] configuration respectively. It is shown that CT_1 with [10,1,1] configuration gives the best effectiveness.

In conclusion, tuning of classification integrated score function should be set differently according to characteristics of messages in environment. Especially, it is better to let content feature take the most portion of effect.

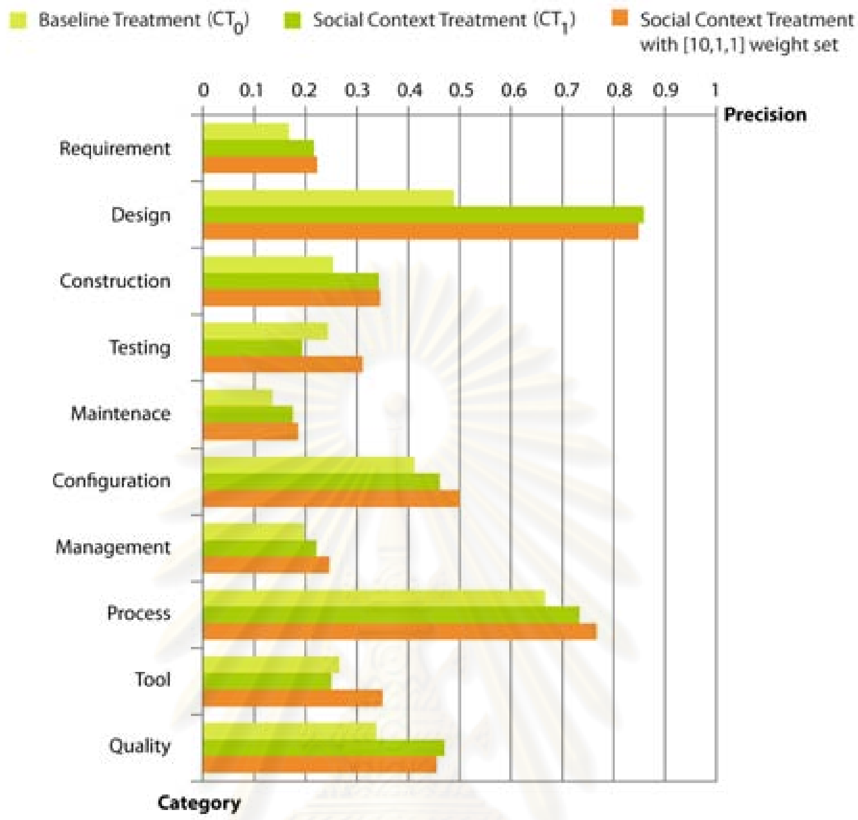


Figure 4.15: Precision comparison of CT_0, CT_1 and CT_1 that is configured with [10,1,1] weight set.

4.3.6.4 Classifier Expansion's Effect

Although our experiment reports that classifier expansion gives better classification effectiveness, we still believe that it does not always provide the positive effect.

Classifier expansion may drop message score down. Whenever a new term is added to a classifier, weights of all terms are recalculated. Other terms will get their weights dropped from the weight normalization process as the size of term vector is increased. When message is parsed to this classifier, there are two possible cases that can occur: the parsed message may contain new terms or may not. For the first case, the message score may be raised up. However, in the second case, the message score will be dropped totally.

For the term that occurs in many categories, even its weight is decreased by the IDF factor,

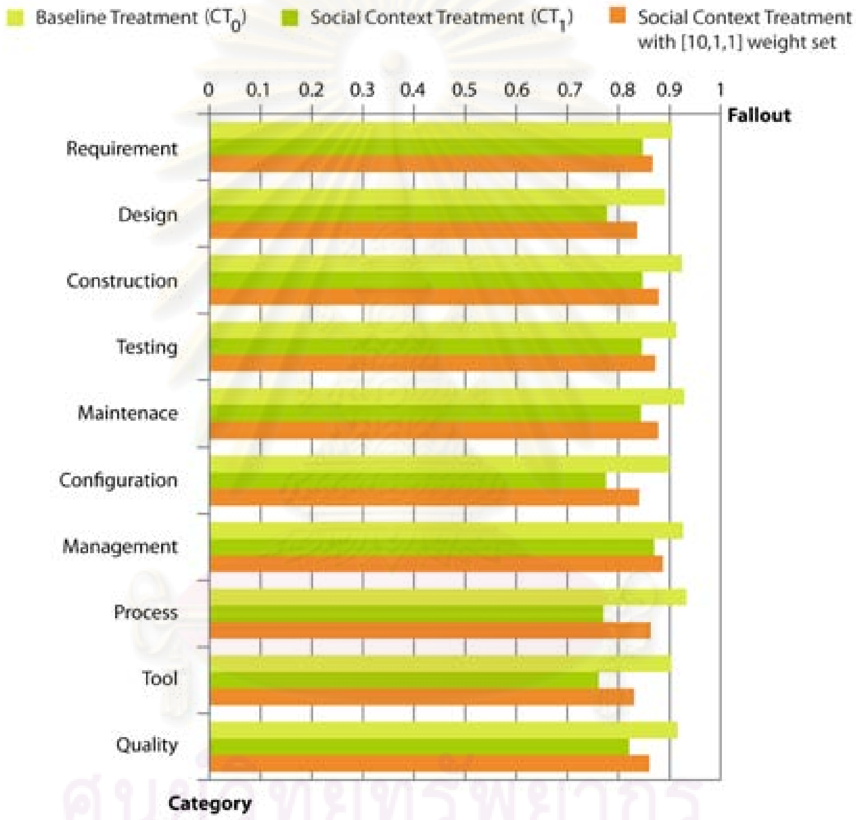


Figure 4.16: Fallout comparison of CT_0, CT_1 and CT_1 that is configured with [10,1,1] weight set.

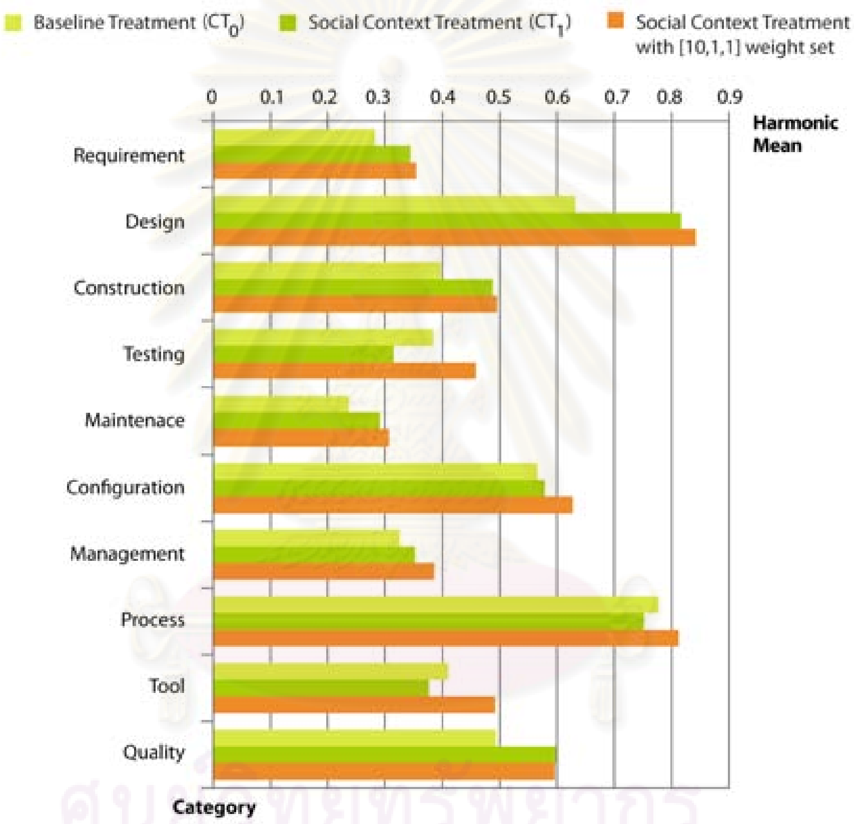


Figure 4.17: Harmonic mean comparison of CT_0, CT_1 and CT_1 that is configured with [10,1,1] weight set.

when its frequency is increased until specific level, it will be the outlier that raises the score of the message higher than it should be. Thus, classifier expansion process should cut these outlier terms off.

As a result from classifier expansion effect, the original classifier should be opened for modification. According to our proposed classifier expansion process, the classifier terms are closed from term frequency modification which makes their frequencies fixed. Thus, when the weight is recalculated, those term weights are slightly dropped. We prohibit the term frequency modification of the original classifier in our approach as we want to keep the original classifier consistent.

4.3.6.5 Classifier Refresh Rate

According to the classifier expansion process described in section 3.6.2, when new term is added to a classifier, weight of this term, and also weight of all terms in this classifier must be recalculated, i.e., a classifier must be refreshed. However, there is an uncertainty about when a new term should be found. In the worst case, a classifier may be refreshed every time each message is parsed. This is practically expensive. Therefore, it is better to predefine the refresh rate, i.e., how many messages to be processed before the classifier is refreshed.

In our classification evaluation experiment, we define the refresh rate in term of MPR and set it to 400. This means that classifiers are refreshed every time after 400 messages are parsed. This number is preliminary picked manually. However, the additional experiment is conducted to monitor the effect of different MPR over CT_2 treatment.

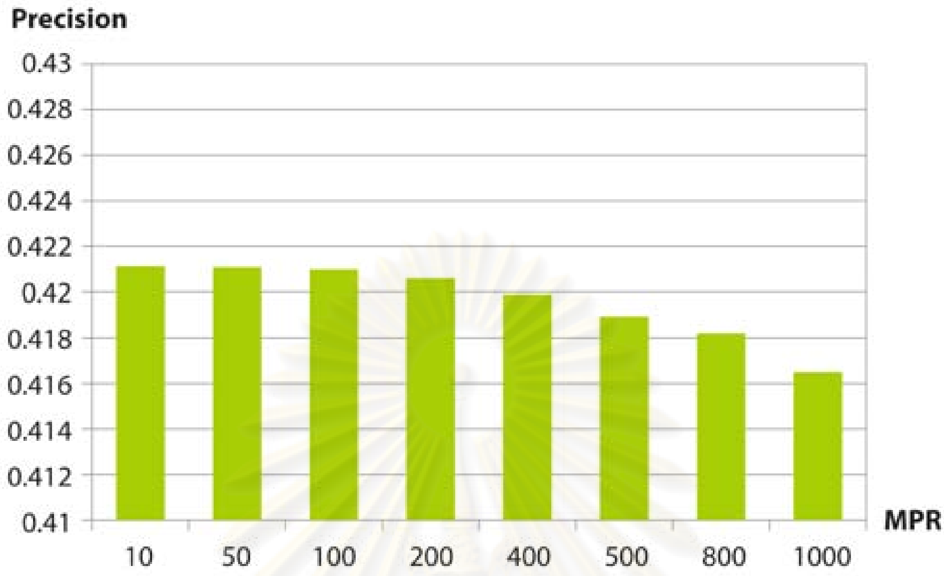


Figure 4.18: Precision comparison among different MPR settings.

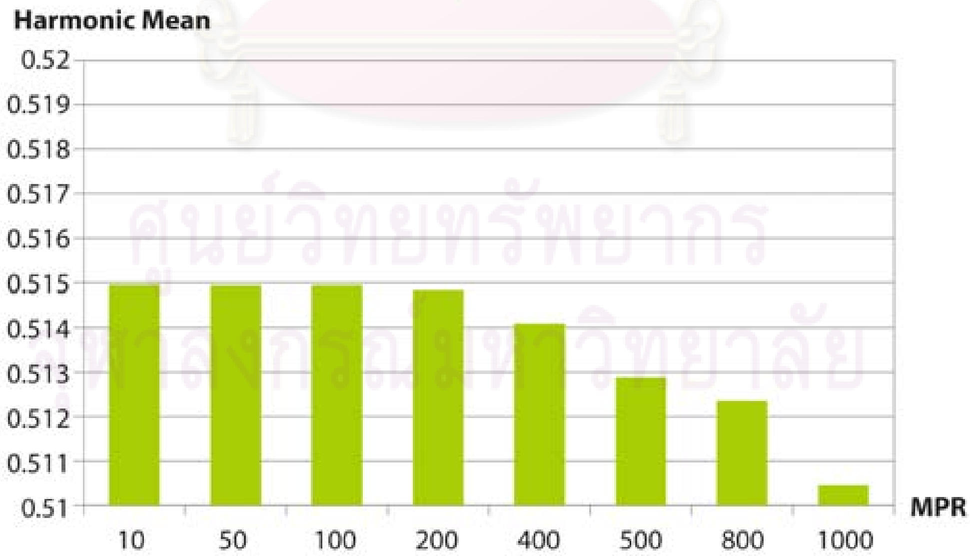


Figure 4.19: Harmonic mean comparison among different MPR settings.

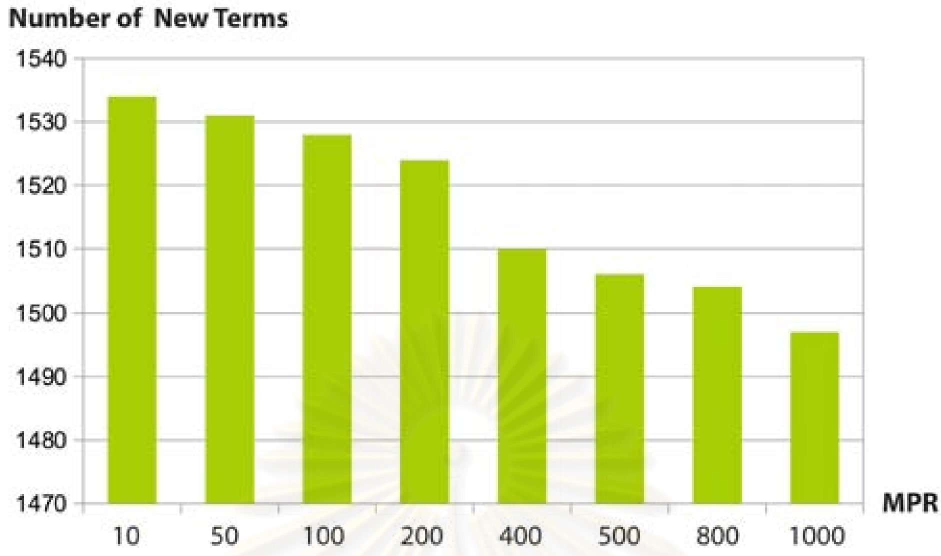


Figure 4.20: Number of new term comparison among different MPR setting.

Eight MPR values: 10, 50, 100, 200, 400, 500, 800 and 1000, are selected. Then, we perform the classification evaluation experiment with CT_2 that is varied with these MPR values. Figure 4.18 and Figure 4.19 shows the precision and harmonic mean of each combination. The results show that when MPR is being increased, precision and harmonic mean are slightly dropped. We also monitor the number of new term added to the classifier and found the same effect as shown in Figure 4.20.

4.4 Retrieval Evaluation

The retrieval evaluation process is straight-forward as shown in Figure 4.21. Firstly, queries are generated from the messages in M_p as described in section 4.2.2. Next, we submit the query to RT_0 and RT_1 treatment. Then, relevances of the result returned from each treatment is evaluated. The judgement on the relevance of message is done based on it usefulness. Finally, effectiveness of both treatments are compared.

4.4.1 Environment

The main control factors of retrieval evaluation are listed in Table 4.6. Total number of query is set to 50. The retrieval will be done over the message in M_e group. We set various document cutoff value r to 5, 10, and 20. There are two treatments used in this evaluation: the

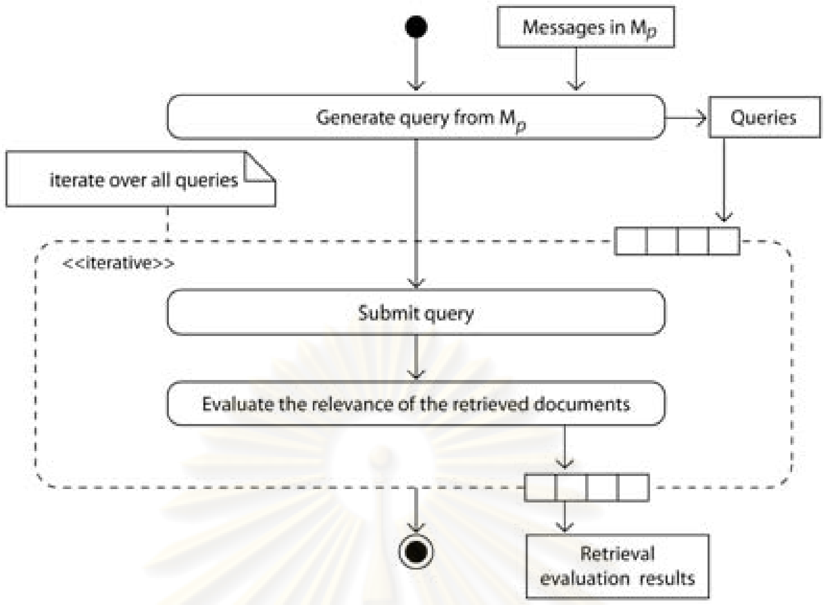


Figure 4.21: Activity diagram of retrieval evaluation process.

baseline treatment, RT_0, which has corresponding weight set $[\omega_{s_1}, \omega_{s_2}, \omega_{s_3}] = [1, 0, 0]$ and the social context treatment, RT_1, which has corresponding weight set $[\omega_{s_1}, \omega_{s_2}, \omega_{s_3}] = [1, 1, 1]$.

Table 4.6: Control factors for retrieval evaluation.

Control Factor	Description	Value
$Q = \{q_0, q_1, ...q_{49}\}$	The set of query used in retrieval evaluation.	Please see Appendix C for full list of query.
M_e	The messages used for retrieval evaluation.	$\parallel M_e \parallel = 12,842$
r	The document cutoff value used in WPR and DCG calculation.	$\{5, 10, 20\}$

4.4.2 Experimental Tool

To support the retrieval evaluation experiment, the retrieval evaluation tool is created. This tool is implemented with Java and Apache Lucene. Its architecture is depicted by Figure 4.22. Messages in M_e are stored in the file system which is done by File System layer. Lucene layer is the interface layer that provides the access to the stored data. Data Model layer is the wrapper layer that maps the stored data to objects and Message Retrieval layer retrieves the message according to a query. This tool has the user interface that shows the retrieved messages. The usage of the tool and its input and output are depicted in 4.23. Figure 4.24 shows an example of this tool. The experimenter has to submit the query and the number of document cutoff (r). The result according to the submitted query will be shown on two sides of the screen. The left side is the results from RT_0 and the right side is from RT_1. The experimenter, then, has to evaluate the relevance of each message on each side by checking at the relevance check box.

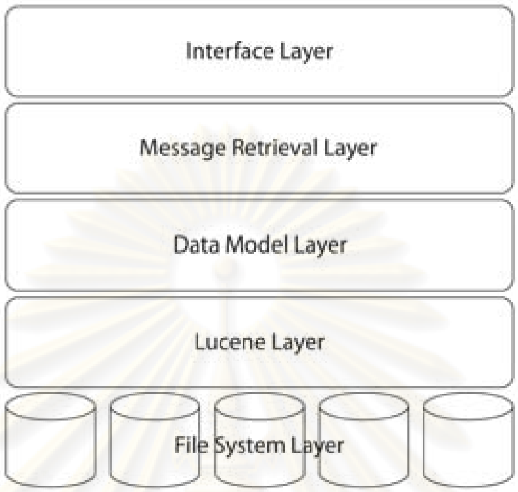


Figure 4.22: Retrieval evaluation tool architecture.

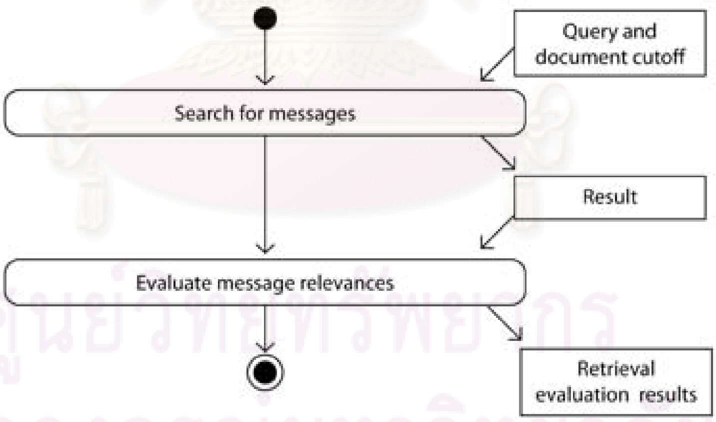


Figure 4.23: Retrieval evaluation tool usage with input and output.

4.4.3 Experimental Result

Figure 4.25 shows the average WPR@5, WPR@10, and WPR@20 of baseline and social context treatment. It indicates that social context treatment gives better WPR in every document cutoff values. The average WPR of this treatment gets highest at the lowest document cutoff,

Figure 4.24: Screenshot of the retrieval evaluation tool when the query ‘javafx’ is submitted.

and gets slightly decreased when document cutoff value is increased. On the other hand, WPR of baseline treatment is increased when document cutoff value gets higher.

Figure 4.26 shows the average DCG@5, DCG@10, and DCG@20 of baseline and social context treatment. It also reports that social context treatment gives better result. However, to statistically state this, the hypothesis testing is needed, which we will go for it in the next section. Full retrieval scores of both treatments can be found in Appendix E.

4.4.4 Experimental Result Analysis

From the results reported in the previous section, we use the statistical analysis to confirm our hypothesis. For retrieval evaluation, we make the hypothesis that social context treatment, RT_1, has better retrieval effectiveness than baseline treatment, RT_0. Thus, null hypothesis and alternative hypothesis are defined as follows.

$$\begin{aligned} H_0 &: \mu_0 \geq \mu_1 \\ H_1 &: \mu_0 < \mu_1 \end{aligned} \tag{4.9}$$

We want to conduct the hypothesis testing based on both WPR and DCG at various document cutoff values. Therefore, μ is defined as a mean of either WPR or DCG at $r \in \{5, 10, 20\}$ where μ_0 belongs to baseline treatment, RT_0, and μ_1 belongs to social context treatment, RT_1.

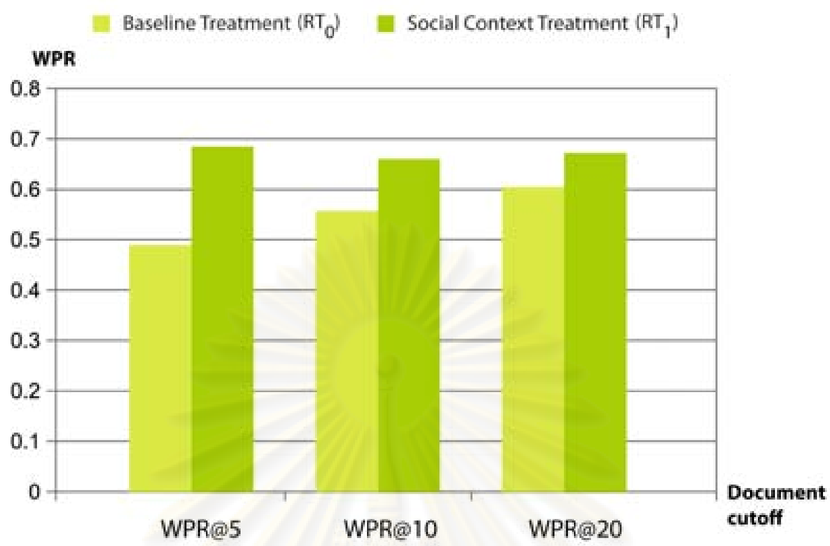


Figure 4.25: Averages WPR@r comparison between RT_0 and RT_1.

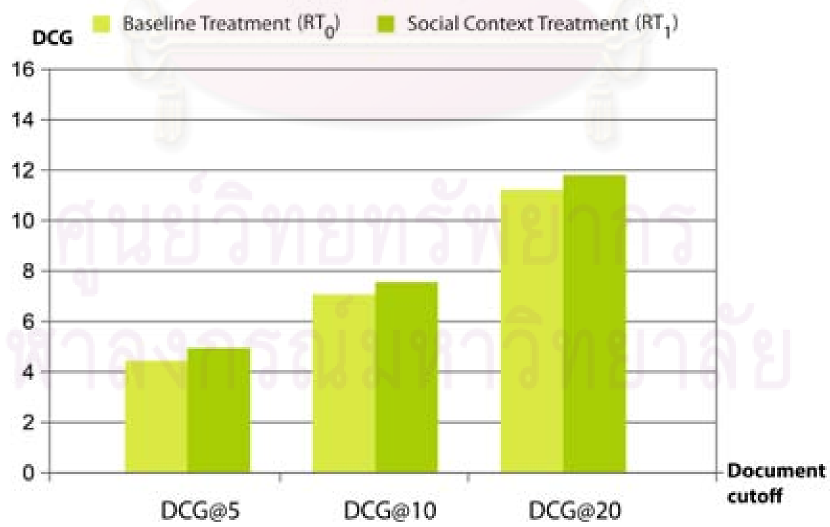


Figure 4.26: Averages DCG@r comparison between RT_0 and RT_1.

Welch Two Sample t-test is selected for this test as the data (the queried messages) are independent for each treatment and the size of sample can be assumed for normality (number of query is 50 which is greater than 30). Firstly, we conduct the test with WPR. The result is shown in Table4.7. Next, the test is performed with DCG as its result is shown in Table 4.8. Both test are 0.05 significant level. The alternative hypothesis at $r \in \{5, 10\}$ are accepted for WPR while those for DCG are all accepted.

Table 4.7: Hypothesis testing result over WPR

Document Cutoff	T	Df	P-value	H_1
5	3.086000	95.484	0.001327	Accepted
10	1.880000	97.021	0.031560	Accepted
20	1.405100	97.535	0.081580	Rejected

Table 4.8: Hypothesis testing result over DCG

Document Cutoff	T	Df	P-value	H_1
5	2.837600	95.581	0.002775	Accepted
10	2.053800	6.821	0.021350	Accepted
20	1.831100	97.495	0.035070	Accepted

4.4.5 Experimental Result Summary

According to the experimental results in section 4.4.3 and the experimental result analysis in section 4.4.4, the result of the experiments can be summarized as follows.

1. It is statistically confirmed that social context treatment has better retrieval effectiveness than baseline treatment in term of WPR when top five or top ten documents of the result are considered. Therefore, we can conclude that, when giving the relevance score in linear regression order, social context results in the higher retrieval effectiveness for the first five or ten document of the result.
2. It is statistically confirmed that social context treatment has better retrieval effectiveness than baseline treatment in term of DCG when top five, top ten and top twenty documents are considered. Therefore, we can conclude that, when giving the relevance score in logarithmic regression order, social context results in the higher retrieval effectiveness for the first five, or ten, or twenty documents of the result.
3. By taking the test on DCG, it is supported that using social context results in more number of relevance message at the beginning of result set.

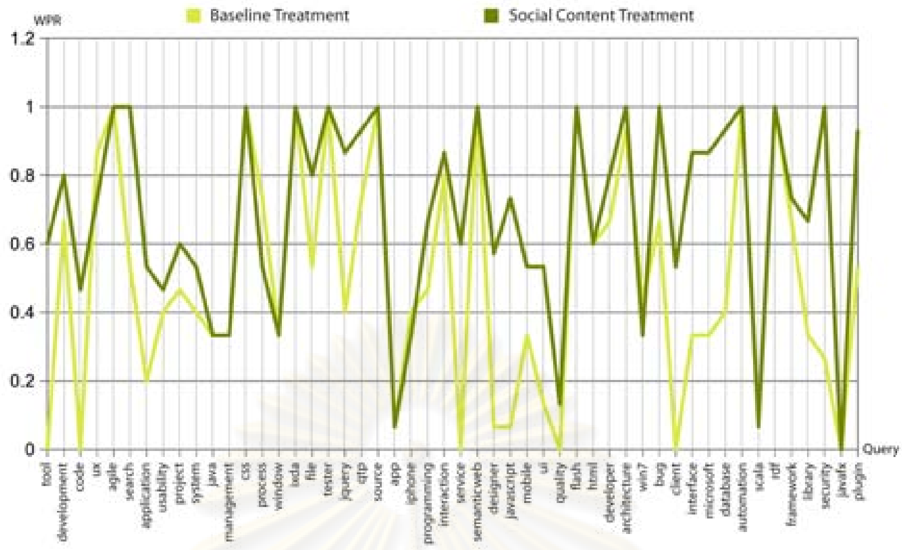


Figure 4.27: WPR@5 comparison between RT_0 and RT_1.

4.4.6 Discussion

According to the results from the retrieval effectiveness evaluation experiment, there are some interesting points for discussion as follows.

4.4.6.1 Effect of Social Context

According to both Figure 4.25 and Figure 4.26, the retrieval effectiveness of baseline treatment gets increased when the document cutoff value is increased while the effectiveness of social context treatment slightly changes. We can imply that the number of relevance messages found by social context treatment at the beginning of result list is greater than those found by baseline treatment. Meanwhile, relevance messages are increasingly found by baseline treatment in the lower rank of the list.

Social context treatment gives better effectiveness in most queries, especially for broad queries. For instance, Figure 4.27 shows the WPR@5 of each queries. We can notice that for broad queries such as ‘application’, ‘file’, ‘service’ and ‘library’, social context gives a significant improvement. These queries, for software engineering related user, are generally used. By applying social context, messages that contains these terms in same context as the information seeker’s interests is ranked higher.

4.4.6.2 Number of Common Friends and Portion of Relevance Message

Social context treatment boosts the scores of the message whose author is more related to the query and is more likely to share same interest with the information seeker. However, it is not guaranteed that message from such author is always relevance. He/she may posts a general message that may hit high rank when it contains a keyword used as query.

We has an assumption that the higher number of common friend the author share to the information seeker, the higher chance he will publish the relevance message. However, Figure 4.28 shows that this is not necessary true. Users who have lower number of common friend also publish relevance messages. Thus, to make the system robust to this situation, additional technique such as Machine Learning should be applied.

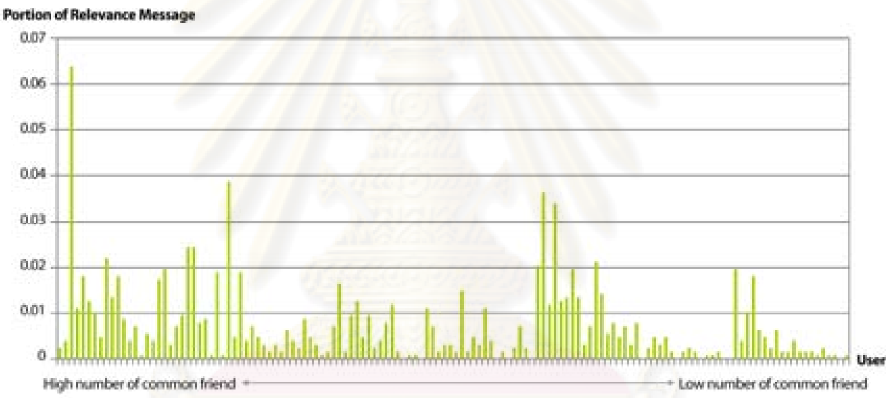


Figure 4.28: Relevance message distribution.

4.4.6.3 Low Retrieval Effectiveness on Some Specific Queries

For specific query, baseline treatments should give high retrieval effectiveness. However, there are some narrow queries such as ‘iphone’, ‘jquery’, and ‘javafx’, that the effectiveness of baseline treatment does not go in the way we expected, nor even for social context treatment. The reason is arbitrary. For example, the messages retrieved from submitting ‘iphone’ as query contains lot of user opinions. The messages from ‘jquery’ contains the mix of user opinion, questions and event. The messages from ‘javafx’ contains lot of questions because the time we collected the data is the period when JavaFX was released (April 2010).

CHAPTER V

TOOL AND IMPLEMENTATION

In this chapter the implementation of the tool named 'SocTweet' is described. Firstly, we begin with the requirements and specifications. Next, the designs of the system including system architecture, system requirement and detailed design are described. Finally, functions of the system are shown with theirs user interfaces.

5.1 Functional Requirements and Specifications

SocTweet is developed with the following functional requirements. Firstly, the system should act as Twitter client that lets user perform basic tasks which are reading messages from his timeline and posting message. The list of friends should be shown and the tool should let user add or remove friends. In addition to these basic functions, the tool must support message classification and retrieval according to the proposed framework. This includes the tasks of adding and removing the classifiers. User should also be able to tune the classification and retrieval by adjusting weights of content feature, user feature, and community feature. These requirements are depicted in Figure 5.1.

5.2 System Design

5.2.1 System Architecture

To implement SocTweet, the system is designed as depicted in Figure 5.2. We use a multi-layer architecture that consists of five layers. Preliminary, Apache Lucene is selected to support the implementation of IR functions such as term indexing and term weighting. It supports various types of storage such as DBMS and file system. In our implementation, we solely select file system storage instead of DBMS so that the tool can be installed by users easily without requiring them to install DBMS on their machines.

File system layer is a physical file system storage that only stores various data. Lucene layer is the integrated part of Lucene. It connects to Lucene and provides the interfaces to perform many IR tasks. This layer also commands File System for data storage and retrieves data according to upper layers' needs. Over Lucene layer, Data Model layer and Communication layer

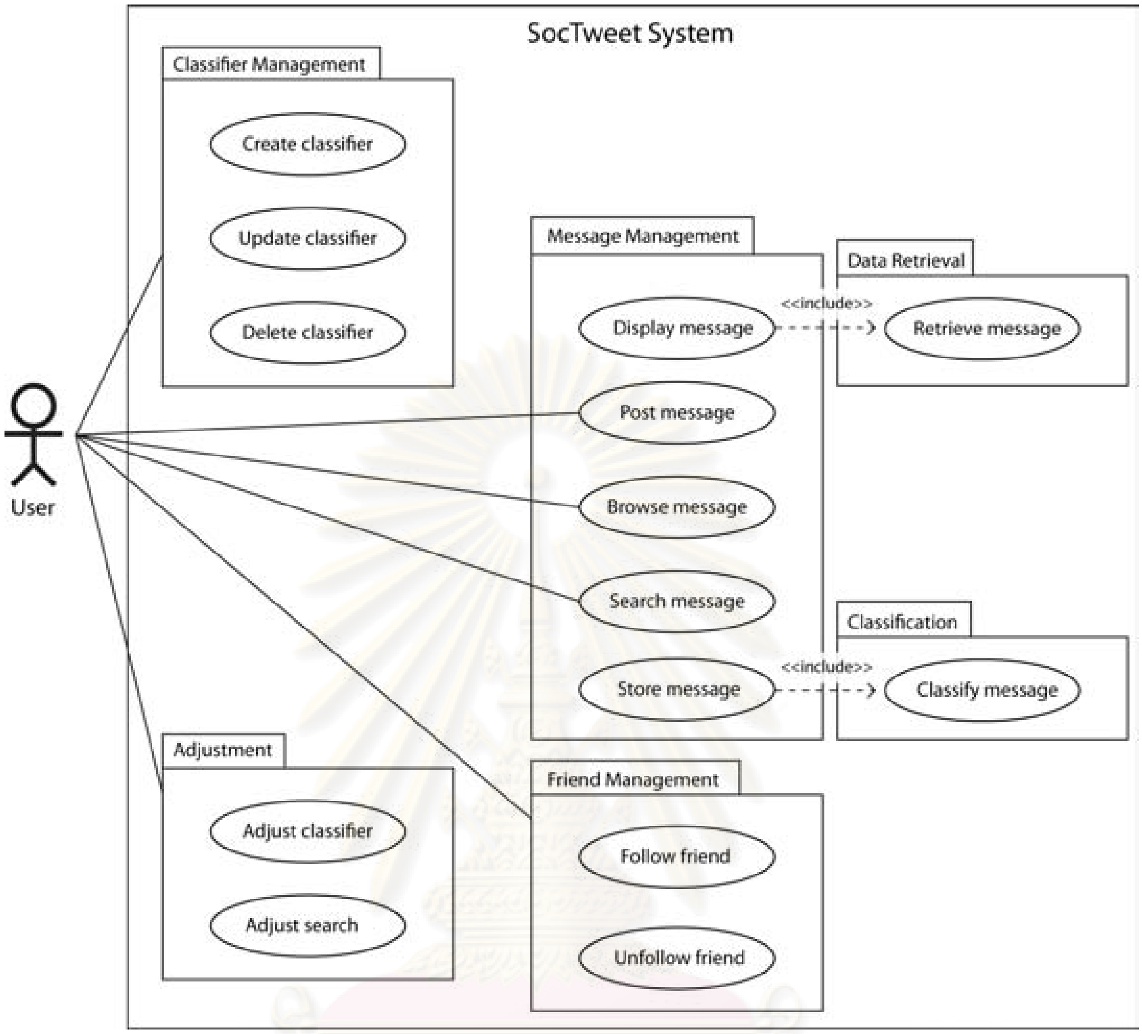


Figure 5.1: Usecase diagram of functional requirements of SocTweet system.

are at their places. Data Model layer implements concept of Object Relational Mapping (ORM) that lets the upper layer access data as if they were objects. Communication layer responds for information sending and receiving from Twitter service. Implements flow of the process, Control layer manages workflow of the system according to business logics. Lastly, on top of all exists Interface layer which presents the data to the users and lets them interact with the entire system.

5.2.2 System Requirement

SocTweet is desktop application implemented with Java and Apache Lucene. It is designed to be platform-independent and only requires user to have Java Virtual Machine (JVM) installed. To use SocTweet, user also needs an account on Twitter and has to grant authentication to the system.

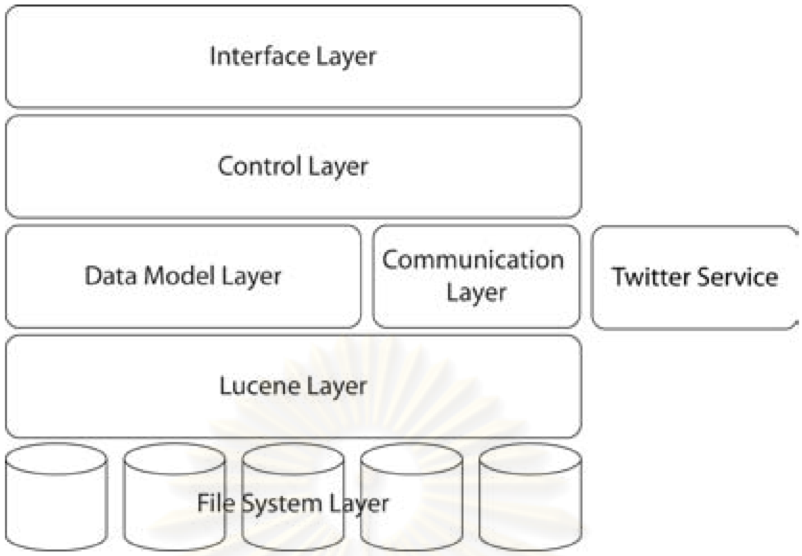


Figure 5.2: SocTweet system architecture.

5.2.3 Entity Classes and Their Relationships

The entity classes in SocTweet and their relationships are depicted in Figure 5.3. Each entity detail is described as follows.

1. **User** is an entity class that represents a system user. Its attributes consist of 'screen_name' (the user name on Twitter), 'twitter_user_id', 'password' and 'access_token' (keep the access token for Twitter service request).
2. **Friend** is an entity class that presents a particular friend of a specific user. Its attributes consist of 'screen_name', 'twitter_user_id', 'profile_image_url', and 'friend_of' (keep the id of the user who is his friend).
3. **Profile** is an entity class that represents a profile, i.e. the messages published in past, of a particular friend. Its attributes consist of 'twitter_user_id', and 'content' (keep the concated content of the message in past).
4. **Message** is an entity class that represents a single Twitter message. Its attributes consist of 'message_id', 'twitter_user_id', 'created_date', 'content', and 'category'.
5. **Classifier** is an entity class that represents a classifier. Its attributes consist of 'title' and 'term_weights' (keep terms and corresponding weights).

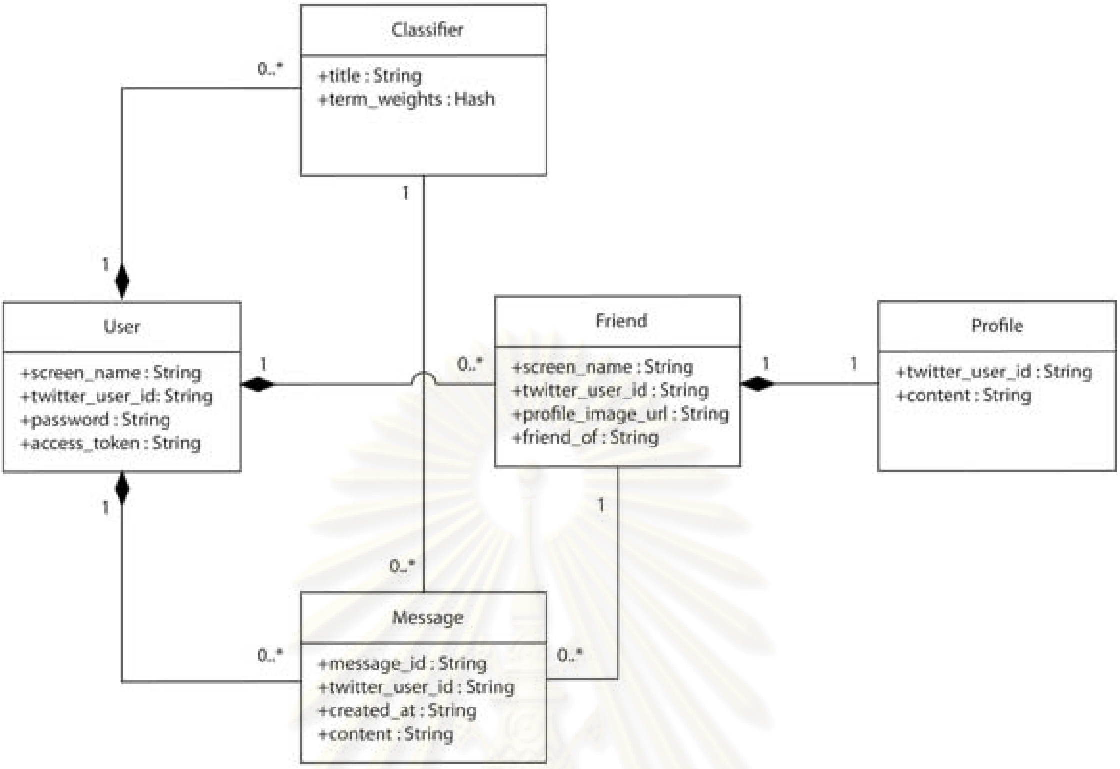


Figure 5.3: Entity classes in SocTweet and their relationships.

5.2.4 Detailed Design

The conceptual design of SocTweet is shown in Figure 5.4. Conceptually, in the lowest layer, there are five storages that store user, friend, profile classifier and message entities. These storage units are controlled by Lucene layer.

Lucene provides basic classes for information storage and retrieval, however there are some complexities to use it directly. IndexWriter and IndexSearcher are two of basic classes that respond to storing and searching the information respectively. Instantiating these two classes requires many parameters and some procedures. We overcome this complexity with Design Pattern by adding Factory classes to the design. IndexWriter is instantiated by IndexWriterFactory. For IndexSearcher, we do not directly apply the Factory class. Instead, its Wrapper class Searcher is created with corresponding Factory class named SearcherFactory. This wrapper is used to simplify the search function.

In Lucene, information is treated as a document. A document contains fields which are defined differently in different type of documents. For instance, Twitter message is treated as a

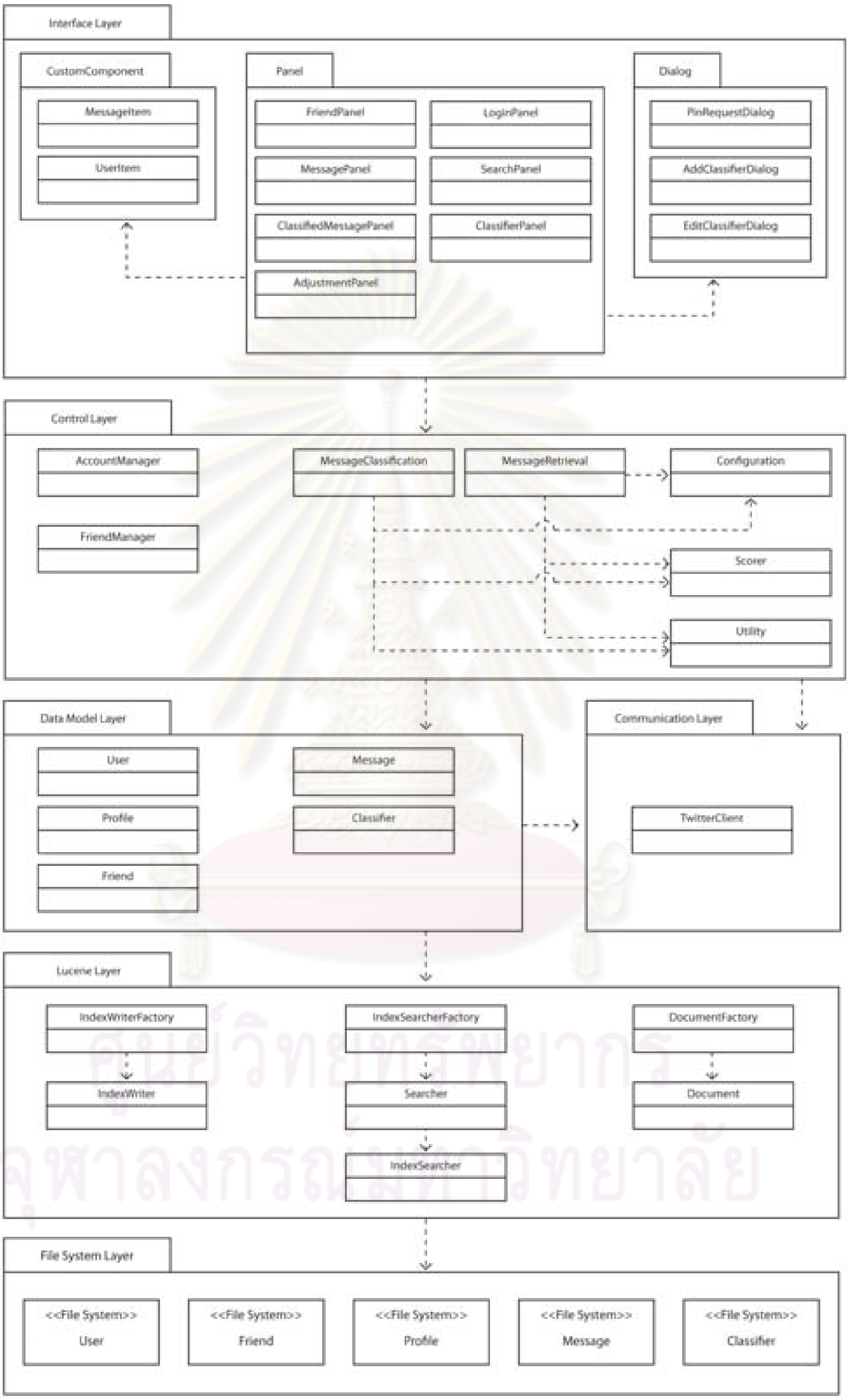


Figure 5.4: Detailed design of SocTweet.

document that has 'message_id' and 'content' fields while friend is also treated as a document that has 'screen_name' and 'twitter_user_id' fields. The Factory class named DocumentFactory is added to the design to help constructing different document type easier and more convenient.

Data model layer contains entity classes that wrap corresponding data stored in File System layer. It provides interfaces that allow data to be accessible as if they were objects. Figure 5.5 depicts the example of this concept. In this example, we want to access information of Friend that has 'twitter_user_id' equal to 2. DocumentFactory requires the type of entity and the key which is 'twitter_user_id'. When this condition is met, it instantiates a Document that has all Friend's attributes from the corresponding storage in File System layer and returns the instance back to the caller. We can then access the attributes of a particular Friend through the dot notation.

Communication layer contains TwitterClient that provides interfaces for Twitter service request. We decide to invoke open source library named Twitter4J for this purpose.

Control layer is the heart of the system. It contains classes that implement business logic according to the requirement. AccountManager takes care of for the authentication of system user. FriendManager handles flow of friend management tasks which are following and removing friend. Utility is a support class that provides some useful functions to others such as string manipulation methods. Configuration records the system preference that is adjustable by the user, for instance, the interval of message update and the weight of features in classification and retrieval tasks. MessageClassification responds for classifying the message according to the configuration defined in Configuration class. MessageRetrieval handles the message retrieval that includes the local message search and remote message retrieve from Twitter service. Both MessageClassification and MessageRetrieval use Scorer class to computed the score for their tasks as described in Chapter 3.

Interface layer contains various classes for interacting with user. We design each screen as panel so that we can reuse them easily. The details regarding the interfaces are described in the next section.

5.3 Functions and User Interfaces

SocTweet is implemented according to the requirements described in section 5.1. The user interface design is made minimally so that the system is simple and easy to use. We divide user

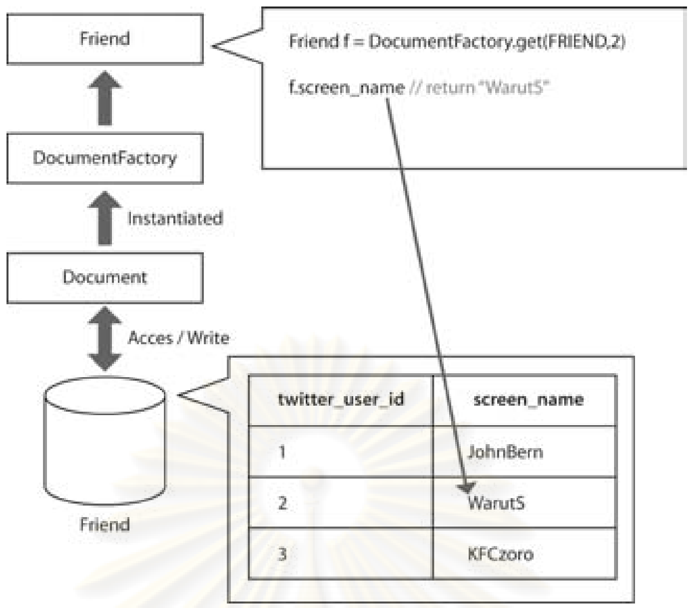


Figure 5.5: Example of entity usage through Document and DocumentFactory.

interface according to its functions. In this section, we firstly described the anatomy of the user interface then follow with the system usages.

5.3.1 User Interface Structure

The main user interface of SocTweet can be divided into three part as shown in 5.6. Labeled as (1) is the main control tab area. The tabs in this areas are divided by their functions as follows.

1. **All Messages tab** responds for listing and posting message.
2. **Classified Messages** responds for listing classified message.
3. **Friends** responds for listing, adding and removing friend.
4. **Search** responds for message searching.
5. **Options** responds for classifier construction and system configuration.

Labeled as (2) is the secondary control component area that changes accordingly to the selected tab on main control tab. Labeled as (3) is the panel area which mainly displays the detail according to the selected tab of main control tab. The next sections guide you through the rest of Soctweet functions and user interfaces.

5.3.2 User Authentication

Before use, user has to sign in to the system. Figure 5.7 shows the sign-in screen. To sign in, user has to input his screen name and password in the text fields and hit the Sign In button. In case that it is the first time of use, user will be prompted with the Pin Request dialog as shown in Figure 5.8. User has to copy the link in the text field in this dialog and open it in the web browser. The page for authorization will be shown and user will be asked to sign in and grant the authorization to SocTweet. After this process, the pin number will be displayed. User must copy this number, replace the link in the text field of Pin Request dialog and press the submit button. If the sign in process failed, the alert dialog will be prompted and the user is required to repeat the sign in step again.

5.3.3 Messages

The basic function of SocTweet is to show updated messages from user's timeline. Figure 5.9 shows the message panel that responds to this task. This panel can be accessed by selecting 'All Messages' tab in main control tab area. The messages are retrieved from Twitter according to the update interval that can be set as described in section 5.3.6.2. At the end of message panel exists the text field that lets the user post the message. This can be done by inputting the desire text and pressing the update button.

Message classification is automatically done as soon as the messages are retrieved from Twitter. To view the classified messages, the user has to select Classified Messages tab in main control tab area. The classified message panel will be shown as depicted in Figure 5.10. The user can select to view the message in each category by selecting the category on secondary tab area.

5.3.4 Friends

SocTweet lets the user manage his friends. To do so, the user has to click at Friends tab in main control tab area. In friend panel, the friend list is shown. To follow a friend, the user has to input the name of that friend in the text field at the bottom of friend panel then clicks at follow button. User can unfollow a friend by clicking at unfollow button at the bottom of each friend block. Figure 5.11 shows the user interface of Friend panel.

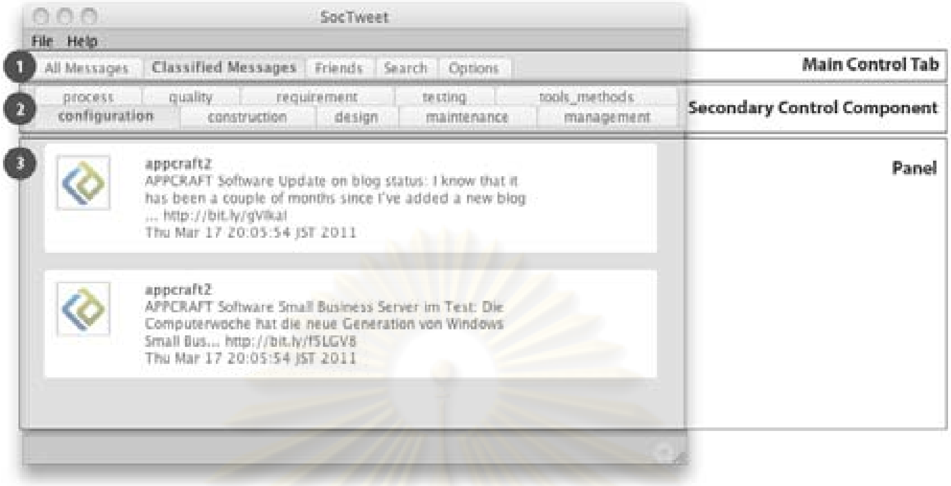


Figure 5.6: SocTweet user interface structure.

Figure 5.7: SocTweet login window.

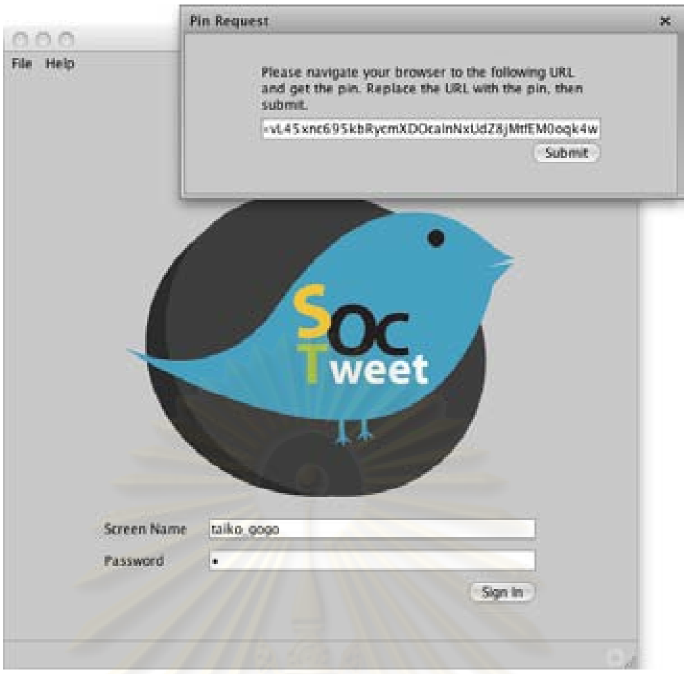


Figure 5.8: The pin request dialog is shown when the user uses the system for the first time.

5.3.5 Search

Old message can be searched in SocTweet. To perform searching, user must access the search panel by clicking at Search tab in main control tab area, input the desire keyword, and click at search button. The search screen is shown in Figure 5.12.

5.3.6 Option

5.3.6.1 Category

The message classification is made according to the category defined in this section. Every time message is retrieved, each category classifies it by comparing relevance according to the classification process described in section 3.6. SocTweet is bundled with 10 categories from SWEBOK knowledge areas. The list of the categories, as shown in Figure 5.13, can be viewed by clicking at Option tab at the main control tab area, then selecting Category tab at the secondary tab area.

Figure 5.14 shows the Add New Category dialog. This dialog allows user to create new category by submitting the category title and text file (in .txt format) that contains some contents

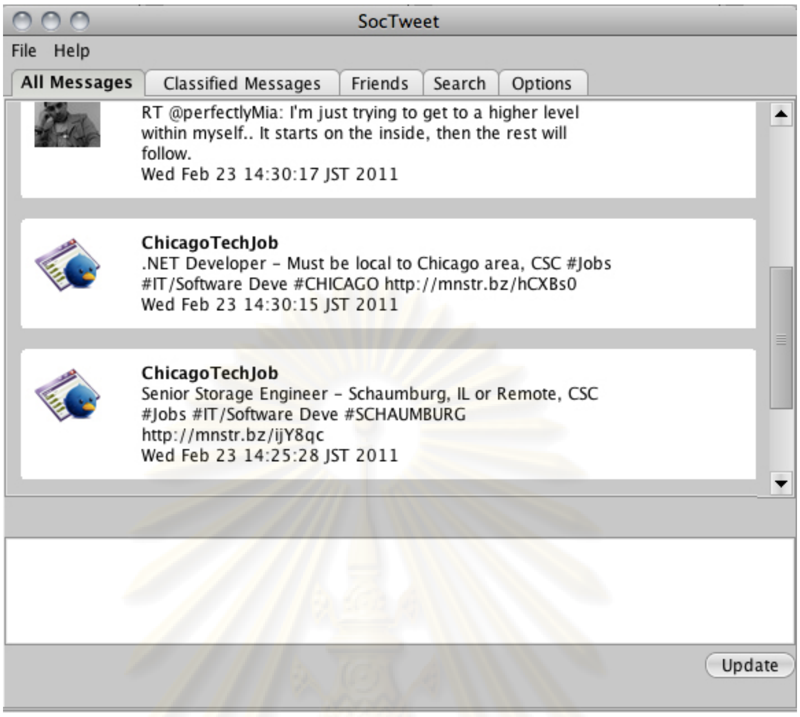


Figure 5.9: Messages are shown in message window.

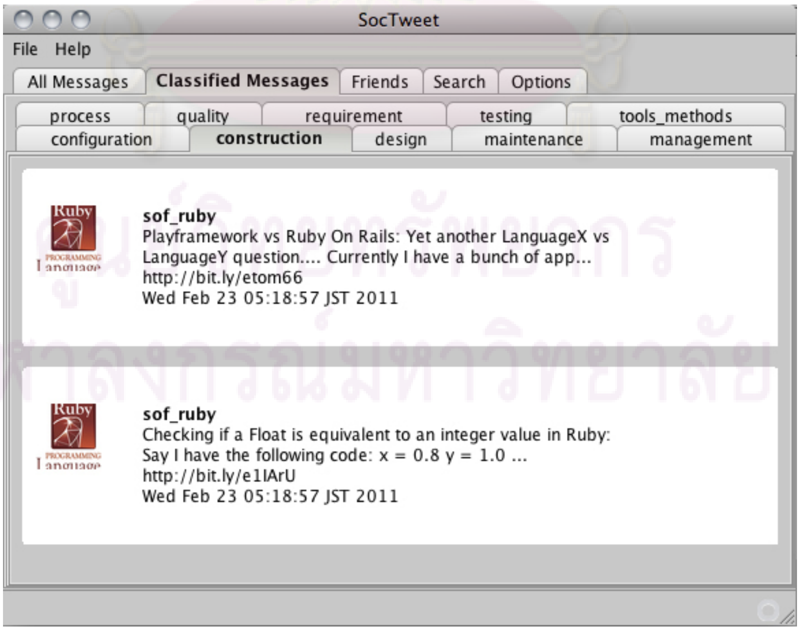


Figure 5.10: Classified message are shown in classified message window.

Figure 5.11: Friend are listed in friend window.

according to the user interests. This dialog can be accessed by clicking at New Category button lying at the bottom of category panel. Category can be edited by resubmitting text file as shown in Figure 5.15.

5.3.6.2 Adjustment

There are some configurations that user can change in Adjustment panel. This panel, as shown in Figure 5.16, can be accessed by clicking at Option tab at the main control tab area then selecting Adjustment tab at the secondary tab area. By clicking at the check boxes, the user can select whether he wants to enable User Profile (user feature) and Link (community feature) in message classification and retrieval. In case of one or both of them are used, the user can set the weight of each factor directly. SocTweet, however, has these option preset. The user only need to adjust these factors if only the preset was not good enough. The user can also set the update interval that controls how often the message will be retrieved.

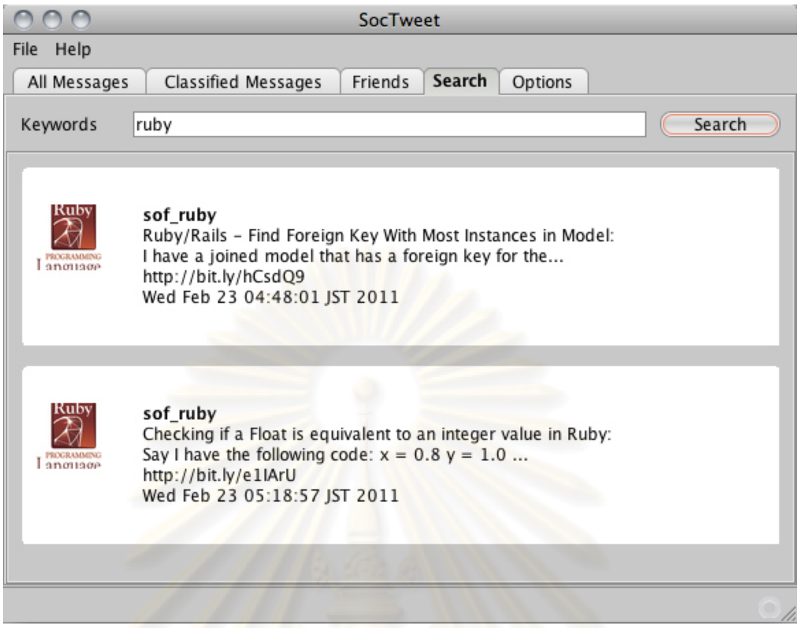


Figure 5.12: Search page lets the user searches for the messages he needs.

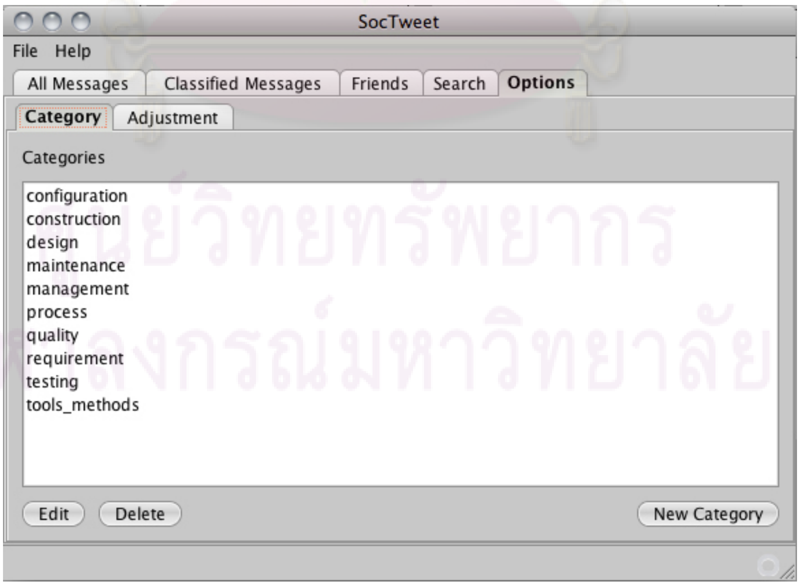


Figure 5.13: List of classifiers are shown in classifier list window.

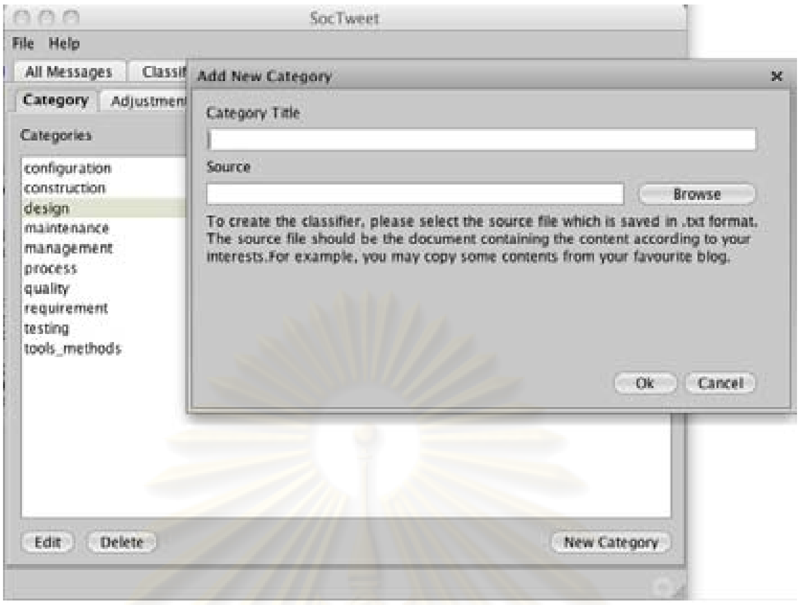


Figure 5.14: User can add new classifier by clicking New Category button and select the text file that contains content of his interests.

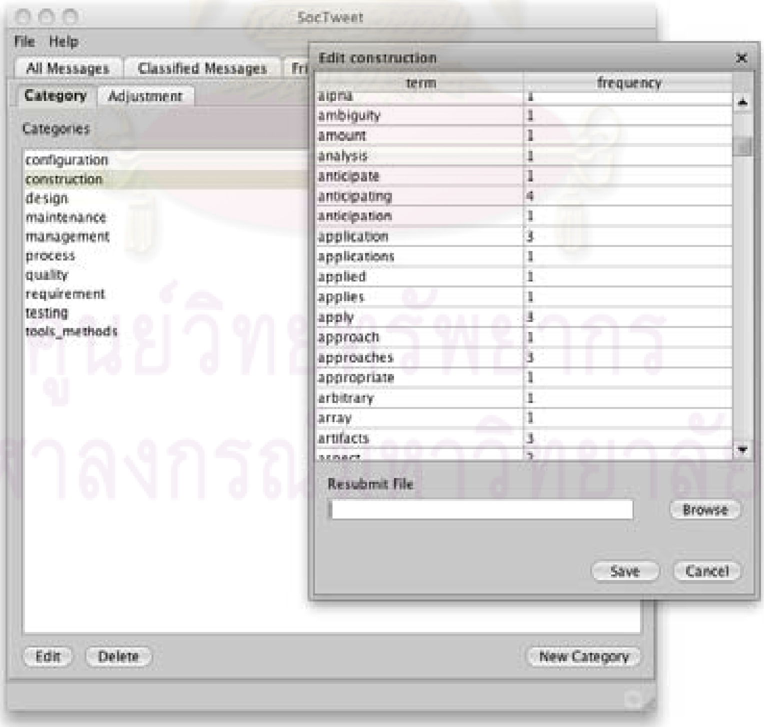


Figure 5.15: User can edit existing classifier by clicking and the classifier title followed with Edit button.

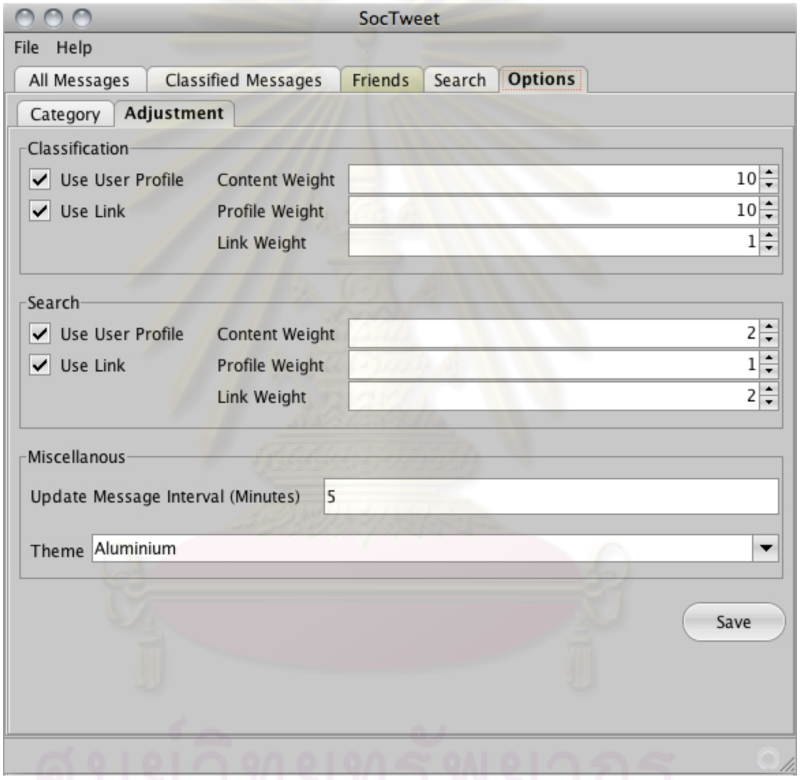


Figure 5.16: Configuration for classification and retrieval task can be done in option window.

CHAPTER VI

RESEARCH SUMMARY

6.1 Research Summary

In this research, we propose the framework and metrics that helps information seeker store and retrieve software engineering related message from Microblogging application. Term-frequency based message classifiers are constructed from SWEBOK. Following the approach of classic Information Retrieval, these classifiers classify message by comparing the textual similarity, i.e, assessing message's content feature. However, due to the characteristics of Microblogging message, solely comparing the content is not much effective. Social context which considers the user feature and community feature of message is combined with the content feature to increase the effectiveness of both message classification and retrieval. User feature assesses similarity between a message author's profile and a classifier. Meanwhile, community feature assesses an impact of message author in two perspectives: an impact among friends in the network and an impact toward the information seeker. From these three feature, the classification integrated score and the retrieval integrated score can be calculated. A message is classified or retrieved when its integrated score exceeds the predefined acceptance threshold.

Not only the social context, we also propose the classifier expansion process that helps increase the classification capability. Whenever a message is added, terms in that message are gathered and are stored in cache. The importance of each term is collected according to the impact of the message authors until it exceeds the predefined term score threshold. The term which its importance exceeds the threshold is added to the classifier and its weight is assigned. Thus, the classifiers can classify message better.

To support what we proposed, the experiments to evaluate the classification and retrieval effectiveness are conducted. We construct the experimental framework and collect the data from Twitter, the most famous Microblogging application, during March to April of 2010. The collected data set consists of 141 users with 528 subscription relations and 208,167 messages. These messages are divided into two groups. The first group with 190,208 messages is used for profile construction. Another group, containing 12,842 messages, is evaluated for its categories by the expert. It is used in classification evaluation and retrieval evaluation.

Three treatments are defined and are compared in classification evaluation. The first treatment is the baseline treatment that solely uses content feature. The second treatment is the social context treatment that uses content feature and social context for message classification. The last treatment applies content feature, social context and classifier expansion process. The effectiveness of each treatment is judged from the harmonic mean metric, the single value metric that is computed from two fundamental metrics: precision and fallout.

For retrieval evaluation, two treatments are defined. The first treatment is the baseline treatment that uses only content feature. The second treatment is the social context treatment that uses both content feature and social context for message retrieval. The effectiveness of each treatment is judged from WPR@r and DCG@r.

The result from classification evaluation experiment shows that, by average, social context with classifier expansion has the highest effectiveness. Its scores on both precision and harmonic mean are the best in most categories even its fallout is slightly dropped as a trade-off. Social context treatment scores second. The statistical analysis is conducted with paired t-test hypothesis testing. It is statistically confirmed that the effectiveness of the social context with classifier expansion treatment is the highest, followed by the social context treatment and baseline treatment respectively.

Similarly, the results from retrieval evaluation experiment also indicates that the social context treatment yields better retrieval effectiveness than the baseline treatment. WPR and DCG are measured with various document cutoff value varied from 5, 10 and 20. The social context treatment gives the highest score for both WPR and DCG at all document cutoff values. The statistical analysis for hypothesis testing is conducted with Welch Two Sample t-test. It is statistically confirmed that, when the retrieval effectiveness is considered in term of WPR, the social context treatment is more effective than the baseline treatment for the first five or ten retrieved documents. In term of DCG, the social context treatment is more effective than the baseline treatment for the first five or ten or twenty retrieved documents.

We also develop the tool that implements the proposed framework. This tool is bundled with the classifiers constructed from SWEBOK. It helps the information seeker in classifying the message from Microblogging automatically and enables the search capability so that the information seeker can search for the message according to his interests.

In conclusion, by applying the social context and the classifier expansion process, the effectiveness of both message classification and message retrieval increases.

6.2 Limitations

In this research, there are some limitations as follows.

1. Although the criteria for user selection are defined, the user selection procedure is semi-random which may not mimic all the property and quality of the real world user network. As the users that are related to some knowledge areas according to SWEBOK, for example, the users that are related to Software Requirement, Software Configuration Management, and Software Engineering Process are hard to find. The user network that has identical interest can not be collected. Therefore, only a single user network with various user interests is used in our research. Although the real network can also contain various categories of topics, the size of the cluster of friends who share common interest is bigger and the density of members is also higher than the network we collect.
2. Different implementation yields slightly different results. We first implemented the experimental framework with Ruby before moving to Java. We found that the setting, for example, the acceptance threshold, is slightly changed. Therefore, the tuning should be done based on the experimental environment and may be slightly different from those reported in this work.
3. In our experiments, the relevance of message is judged as binary value. If message is relevance, it is scored as 1. On the other hand, if the message is not relevance, it is scored as 0. Using fuzzy value for relevance evaluation will give more accuracy and more prone to bias.

6.3 Future Works

There are many points in this research that can be further researched.

1. The experiment of the proposed framework should be conducted with more users in different domains because the experiment in our work is only conducted with a single, software engineering related, user network.

2. The classification integrated score and retrieval integrated score functions are configured with equal weights of content feature, user feature and community feature. Although we show that they result in higher effectiveness compared to baseline, there exists the combination of weights that results in better effectiveness. This tuning is concerned as an optimization problem that additional techniques should be used for finding the best solution such as neural network and SVM.

3. The classifier expansion process can be improved in many ways. For example, the new term that is found should be kept if only it is a noun which is meaningful to detect the topic of interest than the term with other part of speech. The importance of the term can also be measured across different user networks, as the importance of a particular term in one network may be bias.

4. The importance of the term in Microblogging stream may be high only at a period time. For example, an event name is important at a time it occurs. However, when it ends, the important of this event name should be decreased. Thus, adding the time as another feature of the message should help improve the classification and retrieval effectiveness.

References

Swebok 2004 : The guide to software engineering body of knowledge 2004. 2004. [online] Available from : http://www.computer.org/portal/web/swebok. [2009, Octobor].

List of social networking websites. 2009. [online] Available from : http://en.wikipedia.org/wiki/List_of_social_networking_websites. [2009, November].

Growth of twitter 2009. [online] Available from : http://blog.nielsen.com/nielsenwire/online_mobile/twitters-tweet-smell-of-success. [2009, December]

Tim, F. B., Tseng, A. J., and Xiaodan, S. 2007. Why we twitter: understanding microblogging usage and communities. International Conference on Knowledge Discovery and Data Mining pp. 56–65.

Leysia, P., and Amanda, L. H. 2009. Twitter adoption and use in mass convergence and emergency events. Proceedings of the 6th International ISCRAM Conference.

Richardo, B. Y. and Berthier, R. N. 1999. Modern Information Retrieval. Addison Wesley.

Michael, S. B., Adam, M., David, R. K., and Robert, C. M. 2010. Enhancing directed content sharing on the web. CHI '10: Proceedings of the 28th international conference on Human factors in computing systems, pp. 971–980, New York, NY, USA. ACM. doi: http://doi.acm.org/10.1145/1753326.1753470.

Bernard, J. J., Mimi, Z., Kate, S., and Abdur, C.. 2009. Micro-blogging as online word of mouth branding. pp. 3859–3864. doi: http://doi.acm.org/10.1145/1520340.1520584.

Alexander, B., Maxim, G., Maria, G. 2009. Sifting micro-blogging stream for events of user interest. Annual ACM Conference on Research and Development in Information Retrieval pp. 837–837.

Porter, M.F., and Robertson S.E. 1980. New models in probabilistic information retrieval. British Library Research and Development Report 5587.

Barry, S., Owen, P., Kevin, M. 2009. Using twitter to recommend real-time topical news. ACM Conference On Recommender Systems pp. 385–388.

Mouna, K., Sebastian, M. T., Neumann, R. S., Tom, C. 2008. Efficient top-k querying over social-tagging networks. Annual ACM Conference on Research and Development in Information Retrieval pp. 523–530.

Alejandro, R., et al. 2009. A proposal for a semantic intelligent document repository architecture. Electronics, Robotics and Automotive Mechanics Conference 0:69–75. doi: http://doi.ieeecomputersociety.org/10.1109/CERMA.2009.26.

Gerard, S., and Michael, J. M. 1983. Introduction to Modern Information Retrieval. McGraw-Hill Education.

Badrul, M. S., George, K., Joseph, K., and John, R. 2002. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In 5th International Conference on Computer and Information Technology (ICCIT).

David, A. S., Lyndon, K., and Elizabeth, F. C. 2009. Tweet the debates: understanding community annotation of uncollected sources. pp. 3–10. doi: http://doi.acm.org/10.1145/1631144.1631148.

Ricardo, B. Y., and William B. F. 1992. Information Retrieval Data Structures and Algorithms. Prentice Hall PTR.

Dejin, Z. and Mary, B. R. 2009. How and why people twitter: the role that micro-blogging plays in informal communication at work. pp. 243–252. doi: http://doi.acm.org/10.1145/1531674.1531710.

APPENDICES

APPENDIX A

COLLECTED USERS

Table A.1: The list of collected user in the experiment.

Twitter User ID	Screen Name	Description
1186	chrismessina	Agent of Free Will. I work for Google. http://wiki.factoryjoe.com/140-Character-Bios
12831	mikeyk	JavaScript, Python, & Visualization design at Meebo, Night-time iPhone coder (@crimedesksf), Musician
13412	hornbeck	Director of Product and Services at Basho Technologies
38353	wbruce	Rubyist since 2001, Language Tourist, Graphic Designer.
45733	nickf	User experience professional, owner of Blue Flavor, former editor in chief of Digital Web Magazine
66613	jdrumgoole	CEO and founder CloudSplit.com. Founder, PutPlace.com. Cloud guy. Technology guy on The Right Hook (Newstalk 106). Entrepreneur. Loud mouth.
600123	chronicole	Early adopter, late bloomer
623223	kbrock	Ruby Software Guy
746323	jeffpulver	Technology Anthropologist; Entrepreneur; Early-Stage Seed Investor; story teller, Living in Social Media. Producer of #140conf
790205	chadfowler	author, programmer, teacher, runner(?!), musician, speaker, conference organizer
804692	akshayjava	Scientist, Microsoft Ph.D. UMBC 2008
817141	cwilso	Daddy, Microsoft web guy, photographer, diver, and king of my own domain (which goes from *here* to oh, over *there* somewhere.) In approximately that order.
817540	mhausenblas	Linked Data Researcher
824211	bokardo	(Co-founder @performable) (Publisher @bokardo) (Co-creator @abtests)
849101	jmspool	Thank you for encouraging my behavior!
930061	ginatrapani	Blogger and software developer. Commander in Chief of my one-woman army.
1245801	rgaidot	digital/technology enthusiast
1246421	danbri	Euro-Bristolian, FOAF, ex-W3C, Semantic Web, Web TV widgetarian, weekend freetard.
1294621	kidehen	Founder & CEO, OpenLink Software, An Open Linked Data Enthusiast.
1312861	rhacer	
1546381	graybill	web developer, interaction designer, pickle maker
1657311	jvaleski	
1847381	blowdart	.NET developer, author of Beginning ASP.NET Security, honorary London girl geek, brand new borged softie.

Twitter User ID	Screen Name	Description
2384071	timoreilly	Founder and CEO, O'Reilly Media. Watching the alpha geeks, sharing their stories, helping the future unfold.
2825931	jfix	Currently trying to make an international organisation XML-compliant.
4641021	rww	Follow ReadWriteWeb for the latest in web technology and social media trends.
5562702	oracletechnet	Community Evangelist/Dev Programs Guy at Oracle - my opinions are my own and no one else's
5746452	waltmossberg	Tech Columnist
5749952	blogblog	Fabian Nthe ist als Konzepter, UX-Designer sowie Interface Designer und Developer in den Bereichen Interaktive-Medien und Out-of-Home ttig.
5813312	tav	Founder of the Espians creators of Ampify. Lover, writer, coder, social artist, entrepreneur. Addicted to Nutella and Gauloises. More: http://tav.espians.com
5932682	davidjrice	Freelance technologist, rubyist, surfer, snowboarder and human.
6186692	pragdave	
6367402	adamtanner	I'm an INTP. Good at abstract thought and logic. Bad at caring. Lets talk programming.
7345532	mitja_i	I am librarian, working at ISP. Loving all tech/computers/internet stuff. Trekkie.
7835212	vydra	Father. Husband. Agile software tester. Toolsmith.
8231742	sunmicrosystems	Sun develops the technologies that power the global marketplace & is guided by a singular vision The Network is the Computer.
8526432	wycats	jQuery/Merb/DM FTW
8864512	marick	Agile consultant, dabbler in many things. Dilettante by trade.
9207672	nateabele	Lead developer and chief fanboy of the Lithium project, the light, fast web framework for PHP 5.3. Inefficient things upset me.
11518842	gadgetlab	Gadgets and high-tech hardware from Wired.com.
11998042	LukeInTH	Luke Hubbard: Creative hacker living in bangkok working for a new media agency @codegent. Projects: @twitbooth @startupguidetv @awesomecards
12019742	nikhilk	Software Architect at Microsoft, working on .NET, ASP.NET and Silverlight...
12522762	lucasjellema	Oracle, SOA Suite, Java, AMIS, ACE Director, 1994, ADF, SQL
13088772	uwiger	CTO Erlang Solutions, Ltd
13255932	grantmichaels	CAD/CAM Engineer - Ruby, Erlang, Javascript, Clojure - Photographer, Electronica Producer/DJ
13608812	jlin	Magazine/Media hacker in the making
13951412	chakrit	...
14073553	floydmarinescu	InfoQ.com Guy
14084530	osuosl	News updates from the Oregon State University Open Source Lab!

Twitter User ID	Screen Name	Description
14223716	botanicus	Ruby & Ruby on Rails developer & Merb Developer, author of Rango framework & Pupu package tool for static media stuff.
14270033	cquinn	Programmer Guy, Java Posse member
14281405	M4r14nn4	Ruby/Rails programmer. Addicted maths enthusiast. Challenge lover. Doer.
14296383	jsilverman	i fix broken web apps
14306062	kohsukekawa	
14316971	ktukker	BDM Adobe Systems Benelux — New Media — Creative — Online Video — Publishing — Social Media — Diving
14335160	halvorson	Owner, Brain Traffic, a content strategy consultancy. Author, Content Strategy for the Web. Mom. Minnesotan. Also, sassy.
14345141	IxDA	Interaction Design Association Global Twitter Feed
14359848	VirtueMe	TDD & DDD Youngster
14429713	venkat_s	Programmer, Author, Mentor, Trainer
14436716	hammerdr	Software engineering student at Rose-Hulman Institute of Technology.
14437022	ikai	Developer Relations at Google
14464631	BluePojo	I'm a Software Engineer. Ruby, Vibram Fivefingers, and my wife make me happy.
14541402	mlevchin	entrepreneur (PayPal, Slide), investor (Yelp, etc), l33t h@x0r, cyclist
14569541	puredanger	Back off man. I'm a computer scientist.
14635493	alex_gaynor	Pythonista, Djangonaut, host of DjangoDose, student
14658472	roidrage	Ruby guy, analog photo and Polaroid nerd, renown cupcake connoisseur, coffee geek, and an all around amazing horse. Not on steroids.
14825303	shashivelur	OOP, OOD, #Architecture, #Enterprise #Agile, High Scalability, #SemanticWeb Technologies #OSGi and Cars
15022225	NNgroup	Jakob Nielsen, Don Norman, Tog, and colleagues: user advocates focusing on usability and user experience
15133162	rpjday	Linux (embedded and otherwise), training, courseware, technical writing and editing, working on my Novell CNI.
15192970	thebeaverhousen	Digital PR & Tech Geek
15383800	hungryblank	ruby and opensouce enthusiast
15395410	ctomlin	Marketing, SEO & User Experience Consultant
15579487	JohanBarnard	Technology enthusiast, geek, software developer and 4-dimensional being.
15736190	smashingmag	Vitaly Friedman, editor-in-chief of Smashing-Magazine.com and Noupe.com, online magazines dedicated to designers and developers.
15817820	javajuneau	DBA, Java and Jython Developer, Jython Committer for Website and Docs
15837794	jconfino	

Twitter User ID	Screen Name	Description
15851832	RubyInside	The Ruby Inside blog - news, tips and tutorials for Ruby and Rails developers.
15903390	graemerocher	Grails Project Lead at SpringSource - a division of VMware
16169251	xtensha	Digital Strategist and founder of xtensha former e-Business Advisor for Austrade
16437252	MichaelDMcCray	Husband, Father, software developer, I write aspect oriented software, I think of new ways to do things
16550758	RicRoberts	Founder of Swirrl.com Web Developer at Stardotstar Editor of DailyJS.com Blogger for RubyInside.com
16600153	mikaelgrev	Fighter Pilot and Java Developer Combined. Obviously I believe in chaos theory. And 36h days.
16739757	steveonjava	Agile manager by day, Java hacker by night. Author, speaker, and open-source evangelist.
17151314	IATV	Information Architect, Information Literacy, UX, IxD, User Experience, Usability, Design, Prague, Ginkgo Love
17352472	sambastream	Your Online Software Company
17413602	programmableweb	APIs, mashups and code. Because the world's your programmable oyster.
17467170	ErichGamma	
17530305	javaposse	The Java Posse podcast twitter feed
18055613	TheASF	The Apache Software Foundation
18126664	dgildeh	A Drupal web Geek
18194778	PragmaticAndy	
18918415	koush	I write code. Mostly for Android. Sometimes for Mono. For fun.
19038780	kasurot	26 year old male. I work in the IT industry for the local school district. Hobbies: pool, movies, video games, expanding my horizons (learning).
19220550	CMMIAppraiser	Got questions? Get answers!
19362297	aras_p	Lead Graphics Programmer at Unity. I cook code that makes pixels.
19629072	DavidBatty	28 years as a Software Developer/Owner Of A Software Company, IT Trainer, Online since 1987, Web TV Presenter, Public Speaker on Web Marketing & An Accordionist
19846836	kbaribeau	Software Craftsman/Codesmith/Artisan, amateur musician, casual gamer
20306354	jbasilio	Husband, Father of 6, Software Development Geek (C#, F#, SQL Server, ASP.NET, jQuery, Ruby, Python, Haskell, Scheme, Erlang), overall knowledge enthusiast.
20536157	google	News and updates from Google
20941662	JEG2	The Okie Rubyist
20946796	satnamsingh	Computer geek.
21110858	asbradbury	PhD student at the University of Cambridge Computer Laboratory
21128486	IanSommerville	Professor, Software Engineer, Foodie. Interested in socio-technical systems and the problems of enterprise software engineering

Twitter User ID	Screen Name	Description
21457289	MSFTResearch	Microsoft Research is dedicated to conducting both basic and applied research in computer science and software engineering.
22174750	smithrobs	I eat, I code, I (verb) (noun). I sleep.
22398002	praxagora	Editor, community activist. Specializing in open source and software engineering at O'Reilly, also write about policy.
23971403	adriancolyer	CTO of SpringSource, and amateur bike rider
25733176	Shiroginne	tags: Mac's,rails,ruby,objective-c,snow/skate-board,en/ru/jp,death-metal,cyberpunk,capoeira
25981250	TigerHasse	Software Simian, an MCPD and software architect who enjoys programming and also works as an MCT, hooked on F#, functional programming, WPF, Surface.
26207697	piotrgega	Student, Freelancer, Open Source projects supporter (dataobjects,...)
28524327	rsharath	
30369946	wndxlori	Software Architect, Rails developer, Gadget Geek, Dog Lover
34778769	springrod	Creator of Spring, CEO at SpringSource, Author
39219215	micmos	
40896402	brywilliams	Consultant at CityTech Inc. and co-founder of Chicago Groovy User Group
45297725	tharunpkarun	
47366813	basecampnews	News about Basecamp.
49539681	BasilBThoppil	
49725381	garbeam	Open source hacker and professional software developer
50393960	BillGates	Sharing cool things I'm learning through my foundation work and other interests...
51546468	joeerl	Grumpy old man who is neither old nor grumpy
52393480	richardfoote	Oracle DBA, David Bowie fan and all round nice guy ...
57615111	abhi_24_88	
59531743	_J_N_	Software tester, blogger, tweeter, facebooker, farmer, wikipedia editor, orkut hater but buzzer, googler, youtube watcher... durrr, burrr...
59752703	ajay184f	A software tester passionate to learn to test any software
61135090	joshbloch	Effective Java author, API Designer, Swell guy
65080914	ilkerde	Make it simple, but not simpler!
67065339	jon677	I love data mining, social networks, machine learning, business intelligence, pattern recognition, and natural language processing.
72254300	kssreeram	Programming language designer.
73859838	TestingNews	Get News and Articles about Software Testing and Test Automation using HP QTP (Quick Test Professional)
81129050	sdt_intel	Intel software architect working on high-level parallel programming. Views expressed here are my own, not necessarily Intel's.
82305761	ivojto	Web Mage

Twitter User ID	Screen Name	Description
82954292	lanettecream	Software tester, writer, presenter.
83900804	michaelmccool	calligrapher; engineer; computer scientist; professor; entrepreneur; now software architect
84858063	vpenela	Lab Rat
91333167	climagic	Cool Unix/Linux Command Line tricks you can use in 140 characters or less.
93113902	OOLua	OOLua is a test driven, cross platform, non intrusive code generator framework for binding C++ and Lua code
93957809	ericschmidt	CEO Google
104042911	Heriny	hi, there :) I am a network engineer specially interested in network security. /CCIE/PMP/am a CERT
113166944	tntomos	ALM Simplified for $1.61/day! Create & manage everything your team needs for your software development process in a single place!
113713261	ChromiumDev	News and announcements for developers from the Google Chrome team.

APPENDIX B

EXAMPLES OF MESSAGE

Table B.1: Example of messages in Software Requirement category.

Message ID	Content	Author
10901325631	More on the Where 2.0 trend: How the Fashion Industry Uses Location-Based Marketing http://bit.ly/chwgwC	timoreilly
7631315834	OpenID to start playing in the big leagues? Chris Messina seems to think so: http://bit.ly/8d3ec9	vpenela
7131133961	everyone considering using paypal to accept payments should read this story http://bit.ly/6C4oJ7 my personal experience is exactly the same	hungryblank
13412828071	New Topic: Why is Drupal is a Good Choice for a Community Website? http://bit.ly/9GhJlD	IxDA
11084671370	HTML5 microdata http://icio.us/23wieg	micmos
12093481501	What's Next For Mobile Apps? http://bit.ly/di2Mcg	rww
11263305958	New blog post (long): State of the Internet Operating System http://oreil.ly/cyFhMZ My take on the new platform wars, part 1	timoreilly
11263305958	New blog post (long): State of the Internet Operating System http://oreil.ly/cyFhMZ My take on the new platform wars, part 1	timoreilly
13342267123	Good read: Whats Up With Social Objects? http://bit.ly/aZ83w8 (johnnyholland.org)	IATV
11048315398	Android to be bigger than the iPhone by year end? http://feedproxy.google.com/ r/PlanetAndroidCom	thebeaverhousen
11983042214	Multilingualization Testing,What if the application has functionality that wasn't in the requirements?: http://bit.ly/bQNjwx	TestingNews

Table B.2: Example of messages in Software Design category.

Message ID	Content	Author
7190880472	http://drp.ly/86J5r How to turn your rails site into an OAuth Provider #rails #oauth	ivojto
4391050143	Next track: The Architecture of Fun: Emotion, Interaction & Design For Massively Social Games #euroia	blogblog
7280776412	worth reading iPhone Human Interface Principles: Creating a Great User Interface http://bit.ly/4UnmtT	blogblog
11606042376	Functional Programming in object oriented languages. Interesting post: http://bit.ly/aD7juK	jbasilio
9167235740	I prefer non-static methods over static methods even when I have no state, but have trouble explaining why. Am I wrong?	kbaribeau
11355210380	studying osx predicate query mechanism - http://bit.ly/G4blMo	danbri
11898169085	The Myth of Design Limitations - http://bit.ly/9aUh2B	smashingmag
11102268549	Supermodel: ActiveModel-Powered Simple In-Memory Database http://bit.ly/9WGlNo	RubyInside
6337734511	Pancake: How To Stack and Loosely Couple Rack-Based Webapps Together http://bit.ly/4yO3Nb	RubyInside
9691629550	Reading ‘Designing Web Interfaces’ http://bit.ly/bxoOMw - lots of good stuff when thinking about designing #ux for your #ria	nikhilk

Table B.3: Example of messages in Software Construction category.

Message ID	Content	Author
10994697797	EC2/EBS allows you to suspend/resume the OS and only pay for the actual hours used. I can now afford a powerful host in the cloud.	vydra
11977274599	Introduction to Perl: Perl is an powerful and adaptable scripting language. It was developed by Larry Wall, who wa... http://bit.ly/dAOzjz	TestingNews
10369663619	How CSS Sprites helps your websites? http://tinyurl.com/ybrkf2y	BasilBThoppil
8185802332	Develop Twiiter client in php using OAuth_Twitter.php http://www.phpclasses.org/browse/package/5941.html	BasilBThoppil
10315239657	’Pragmatic F# in Action’ with Amanda Laucher and Josh Graham : http://bit.ly/9EosqE #infoq #fsharp	TigerHasse
7426178107	Ruby, Heroku and Cloud Computing - I am really pleased with Heroku, which I havent talked about yet. Its a... http://tumblr.com/xfd59y9u8	jsilverman
8851169981	Since Javascript has lambdas it’s as awesome as ruby when it comes to working with arrays quickly. http://pastie.org/816095	ivojto
2418416411	I’m impressed by Lua. So simple, but powerful. Cool!	botanicus
9163108565	”Compare JavaScript Frameworks” http://bit.ly/ciYjt7	jbasilio
11255781257	searching again for oauth+atompub work, found http://rollerweblogger.org/roller/entry/oauth_for_roller	danbri

Table B.4: Example of messages in Software Testing category.

Message ID	Content	Author
10608252574	QA - Quality Assistance? Interview with Jon Bach on uTest http://bit.ly/bLglDw #softwaretesting #qa #WeekendTesting	ajay184f
11502008524	Then tester does some sanity/regression automation, then really explores the changes. Tests like crazy, and gives the dev + & - feedback.	lanettecream
8596073255	Nice software test plan example: http://bazman.tripod.com/frame.html	vydra
11902730939	Automation Adoption: By William Coleman One of the basic challenges with test automation is adoption. I cant t... http://bit.ly/95qUE3	TestingNews
11977062227	What is the QuickTest Automation Object Model (AOM) and how is it used ?: The QuickTest Professional (QTP) Automat... http://bit.ly/cxevrn	TestingNews
11866156311	A Ground Up Kit for Software Testing — Used Test Equipment: Software Testing: What is Software Testing? There are ... http://bit.ly/9l75n1	tntomos
11368426951	TDD for Embedded C over at PragProg => http://www.pragprog.com/titles/jgade/test-driven-development-for-embedded-c	grantmichaels
9984980722	”Behavior Driven Development (BDD) with SpecFlow and ASP.NET MVC” http://j.mp/aWHRVo	jbasilio
4569489400	Interesting - explaining TDD/BDD via queuing theory: http://jbrains.ca/permalink/285	smithrobs
13730015317	iPad Usability: First Findings From User Testing via Jakob Nielsen’s Alertbox http://bit.ly/9TadXu	ctomlin

Table B.5: Example of messages in Software Maintenance category.

Message ID	Content	Author
8821203828	After using Ubuntu for java dev for several years, I thought moving to a Win64 shop would be a pain, but actually not bad with cygwin.	vydra
11430082764	Configuration & Testing in Preventive Maintenance Software: Web based CMMS Software programs help public and priva... http://bit.ly/bb8m2Z	tntomos
2284776371	I’ve just migrated to Nginx. I love it. But 101ideas.cz is still down, it needs at least quick rewrite.	botanicus
11938123720	intro to nginx.conf scripting => http://agentzh.org/misc/slides/nginx-conf-scripting/nginx-conf-scripting.html#1	grantmichaels
8357942265	Refactoring to Patterns by Kerievsky is amazing. So many great insights and expansions on Fowler’s Refactoring	hammerdr
12014923346	DbKeeperNet 1.1.1.1 (BSD License): A component to help you manage relational database schema. http://bit.ly/9pmyTk	abhi_24_88
9621607491	InfoQ: Facebooks Petabyte Scale Data Warehouse using Hive and Hadoop http://bit.ly/cfq6U8	jbasilio
8591463740	Screencast: How To Upgrade Your Rails 2 App to Rails 3 in 25 Minutes http://bit.ly/9Ira0o	RubyInside
3700555507	looks like IBM and Progress plan on competing with #dmserver and the Spring open source #osgi projects with the Apache Aries proposal.	adriancolyer
5806549062	Experimental OpenSUSE RPM for #hudsonci at http://hudson-ci.org/opensuse/ . Please try it and let me know if it works	kohsukekawa

Table B.6: Example of messages in Software Configuration Management category.

Message ID	Content	Author
11430082764	Configuration & Testing in Preventive Maintenance Software: Web based CMMS Software programs help public and priva... http://bit.ly/bb8m2Z	tntomos
9179880633	Git, kicking it OS X style. http://wiki.github.com/Caged/gitnub/	RicRoberts
7209712471	Search for ”3.7.2” in the source code to get all the changes. Here’s the change log: http://bit.ly/4s8Awq #miglayout	mikaelgrev
7946470938	#git makes switching #grails versions during development so trivial. ie. git co master/1.2.x/1.1.x	graemerocher
8907426063	Subversion vs. Git: Can you feel the desperation? http://subversion.wandisco.com/component/content/article/1/40.html	nateabele
6435512699	interesting book, writing XMPP apps with javascript http://bit.ly/5Ueg4O with code on github http://bit.ly/83LDkj	hungryblank
10991462804	Another Git strategy, using rebase as opposed to –no-ff: http://geewax.org/2009/11/21/agile-git-workflow.html	JEG2
10416578298	http://github.com/uwiger/pots will of course work better...	uwiger
8970422654	Trac + Stickies for project management is painful. Debating Redmine vs retrospectiva http://is.gd/2IGrC	kbrock
11764823726	4 features to make #github an awesome platform http://tav.espians.com/4-features-to-make-github-an-awesome-platform.html	tav

Table B.7: Example of messages in Software Engineering Management category.

Message ID	Content	Author
11905165851	How to Implement QA Process ?: http://qualitypointtech.net/NewsFeed/13115-How-to-Implement-QA-Process-.html	TestingNews
11919804631	How to do Effort Estimation In Software Testing: http://bit.ly/cBmRgU	TestingNews
11430082764	Configuration & Testing in Preventive Maintenance Software: Web based CMMS Software programs help public and priva... http://bit.ly/bb8m2Z	tntomos
11874894804	Dealing With Clients Who Refuse To Pay - http://su.pr/2jAyc5	smashingmag
10518175350	Software engineering from Dilbert. Superb. http://bit.ly/d1ZD06	IanSommerville
4834467612	Increase Your Agency’s Productivity — Yield Software http://ow.ly/15UBGk	basecampnews
5811213563	Retrospectiva: Open Source Project Management Rails App http://bit.ly/1CRWwm	RubyInside
4895783275	The Advantages of Making Decisions with Accurate Information http://bit.ly/1IJE1g	jon677
2278126524	New white paper: An Introduction to Document Management (http://bit.ly/2mhjFw	sambastream
12427092297	Most of the top 10 of BusinessWeek’s ”Most Innovative Companies of 2010” have invested in UX: http://bit.ly/9JR8Fo	nickf

Table B.8: Example of messages in Software Engineering Process category.

Message ID	Content	Author
14644733192	My Peru CMMI conference keynote page is up: http://bit.ly/a7A7kw	CMMIAppraiser
14160890172	CMMI is something you USE. Agile is something you ARE.	CMMIAppraiser
7756316680	I propose for the 2010 #agile improvement that #scrum standup meetings include from everyone ”what did I learn yesterday”	MichaelDMcCray
13905384095	"Mision Impossible: Shrinking the UX Process" http://bit.ly/cta6h5 (uxbooth.com)	IATV
13988249098	TDD enables declaration of intentions as tests ... CMMI+SCAMPI enables TDD for process. Try it, but allow for self subscription and agility.	CMMIAppraiser
1155558198	Think of OPP as the supplier of statistical analysis and data and QPM as the consumer. They do both feed one another also.	CMMIAppraiser
1147100566	GP2.3 (Provide Resources) is about more than just ”people.” It includes hardware, space, sofware, tools, templates, methods, etc...	CMMIAppraiser
1140796483	GP2.1 doesn’t have to be a big policy book. Try posters, training materials, or regular emails for leadership	CMMIAppraiser
1133066675	GP2 2 is useful if you use the process as the foundation for you project plan	CMMIAppraiser
1284131462	The CMMI has MANY usage modes. More than just ”process improvement” the CMMI can meet the needs of customers, management, and practitioners	CMMIAppraiser

Table B.9: Example of messages in Software Engineering Tools and Methods category.

Message ID	Content	Author
10608252574	QA - Quality Assistance? Interview with Jon Bach on uTest http://bit.ly/bLglDw #softwaretesting #qa #WeekendTesting	ajay184f
11939770513	UAT by the QA Team: From what I understood about the question , you were referring to bugs discovered in UAT (and... http://bit.ly/abJvR3	TestingNews
11234833119	Evolution of software testing in India: Testing is one of the final and most important steps in creating a softwar... http://bit.ly/c4taxf	tntomos
11715105299	Quality Assurance & Software Testing by Softage: Software development process is a well http://goo.gl/fb/D99lt	tharunpkarun
11873583029	The Lost Element of Quality - http://bit.ly/cxsqU3 - Interesting read.	smashingmag
4572191591	The 15 Step Rails Code Quality Checklist http://bit.ly/1ye2fp	RubyInside
13563732062	Good usability podcasts worth hearing: Q&A with UX Experts on Usability and Prototyping via UIE http://bit.ly/dD4WYK	ctomlin
13499618717	Great article by David Travis: ”Creative ways to solve usability problems” http://bit.ly/bEQWuG (www.userfocus.co.uk)	IATV
5423877514	Alertbox: Agile User Experience Projects - http://bit.ly/AgileUsability	NNgroup
10304584419	Code Bubbles - Rethinking the programmer’s UI. http://bit.ly/dlJanE	kssreeram

Table B.10: Example of messages in Software Quality category.

Message ID	Content	Author
11899860575	Test Design Studio 2 an updated review: Ive previously reviewed TDS (Test Design Studio) version 1, and I s... http://bit.ly/aPrlOw	TestingNews
10361767261	Visual Studio 2010: Introduction To The New #Architecture #Tools : http://bit.ly/avrHmo #vs2010 #dotnet	TigerHasse
6745725577	JetBrains RubyMine 2.0 - Well, I normally side with the anti-IDE camp when it comes to Ruby, but IntelliJ... http://tumblr.com/xfd4qnt3s	jsilverman
8359414902	Android 2.1 emulator on Fedora 12: http://cli.gs/T2VSP1. Start with just running the emulator. More coming.	rpjday
11932239679	prettyLoader: a small jQuery plugin that displays AJAX loader next to the mouse cursor - http://bit.ly/dam8P8	smashingmag
9934743124	Attn developers: Introducing the Google PowerMeter API http://bit.ly/aNFUqj	google
5450042092	New post: Drupal for Marketing - Google Analytics (http://www.davidgildeh.com/node/172)	dgildeh
7365124825	Groovy-Eclipse tools update to 2.0 RC1 : http://bit.ly/8SlQrT	adriancolyer
4895119460	IntelliJ IDEA is open source. The Java Posse got the scoop in a special episode: http://bit.ly/3czL34	javaposse
9193018457	Looks like Netbeans 6.9 M1 has been released: http://wiki.netbeans.org/NewAndNoteworthy69m1	javajuneau

APPENDIX C

FULL LIST OF QUERY

Table C.1: List of generated query.

ID	Query	Frequency	ID	Query	Frequency
1	tool	576	26	service	221
2	development	443	27	semanticweb	215
3	code	437	28	designer	209
4	ux	435	29	javascript	203
5	agile	393	30	mobile	196
6	search	351	31	ui	189
7	application	348	32	quality	189
8	usability	347	33	flash	167
9	project	331	34	html	167
10	system	327	35	developer	163
11	java	324	36	architecture	162
12	management	308	37	win7	161
13	css	283	38	bug	161
14	process	280	39	client	155
15	window	275	40	interface	150
16	ixda	271	41	microsoft	148
17	file	267	42	database	142
18	tester	263	43	automation	139
19	jquery	256	44	scala	136
20	qtp	256	45	rdf	132
21	source	249	46	framework	131
22	app	236	47	library	130
23	iphone	234	48	security	129
24	programming	230	49	javafx	123
25	interaction	229	50	plugin	123

APPENDIX D

CLASSIFICATION EVALUATION EXPERIMENT RESULTS

D.1 Classification Evaluated Score

Table D.1: True positive correctness, False negative correctness, precision, fallout and harmonic mean of baseline treatment CT_0

Category	TP	FN	Precision	Fallout	Harmonic Mean
Requirement	23	11698	0.3382	0.9158	0.4940
Design	259	10923	0.2534	0.9241	0.3978
Construction	403	9435	0.1671	0.9046	0.2821
Testing	260	11617	0.6667	0.9329	0.7776
Maintenance	27	11739	0.1357	0.9285	0.2368
Configuration	29	11624	0.2437	0.9136	0.3848
Management	75	11395	0.4121	0.9001	0.5653
Process	45	11367	0.4891	0.8915	0.6317
Tool	221	10865	0.1977	0.9267	0.3258
Quality	69	11365	0.2654	0.9033	0.3902
Average	141.1	11202.8	0.3169	0.9141	0.4486

Table D.2: True positive correctness, False negative correctness, precision, fallout and harmonic mean of baseline treatment CT_1

Category	TP	FN	Precision	Fallout	Harmonic Mean
Requirement	32	10490	0.4706	0.8212	0.5983
Design	350	10010	0.3425	0.8469	0.4877
Construction	522	8839	0.2164	0.8475	0.3448
Testing	286	9594	0.7333	0.7705	0.7514
Maintenance	35	10669	0.1759	0.8439	0.2911
Configuration	23	10760	0.1933	0.8457	0.3146
Management	84	9824	0.4615	0.7760	0.5788
Process	79	9912	0.8587	0.7774	0.8160
Tool	247	10199	0.2209	0.8699	0.3524
Quality	65	9579	0.2500	0.7613	0.3764
Average	172.3	9987.6	0.3923	0.8160	0.4912

Table D.3: True positive correctness, False negative correctness, precision, fallout and harmonic mean of baseline treatment CT_2

Category	TP	FN	Precision	Fallout	Harmonic Mean
Requirement	35	10203	0.5147	0.7987	0.6260
Design	383	9776	0.3748	0.8271	0.5158
Construction	575	8571	0.2384	0.8218	0.3696
Testing	283	9156	0.7256	0.7353	0.7304
Maintenance	40	10352	0.2010	0.8188	0.3228
Configuration	26	10478	0.2185	0.8235	0.3454
Management	83	9384	0.4560	0.7412	0.5647
Process	79	9559	0.8587	0.7497	0.8005
Tool	266	9887	0.2379	0.8433	0.3711
Quality	97	9229	0.3731	0.7335	0.4946
Average	186.7	9659.5	0.4199	0.7893	0.5141

Table D.4: True positive correctness, False negative correctness, precision, fallout and harmonic mean of baseline treatment CT_1 with weight set [10,1,1].

Category	TP	FN	Precision	Fallout	Harmonic Mean
Requirement	31	11000	0.4559	0.8611	0.5962
Design	353	10380	0.3454	0.8782	0.4958
Construction	537	9043	0.2226	0.8670	0.3543
Testing	299	10748	0.7667	0.8632	0.8121
Maintenance	37	11098	0.1859	0.8778	0.3069
Configuration	37	11095	0.3109	0.8720	0.4584
Management	91	10640	0.5000	0.8404	0.6270
Process	78	10671	0.8478	0.8369	0.8423
Tool	276	10401	0.2469	0.8872	0.3863
Quality	91	10450	0.3500	0.8306	0.4925
Average	183	10552.6	0.4232	0.8614	0.5372

D.2 Term Occurrence Comparison

Table D.5: Top 50 terms from classifier, classifier and extended terms in Software Requirement category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
requirements	259	web	7	web	100
software	214	iphone	7	blog	90
process	67	os	5	iphone	79
system	37	data	5	google	79
requirement	29	software	5	ipad	75
topic	28	engineering	4	day	71
specification	26	news	4	java	63
engineering	23	linkeddata	4	linkeddata	62
engineer	22	blog	4	app	43
example	22	read	4	post	38
kot00	22	social	4	talk	33
analysis	20	re	3	twitter	33
design	20	papers	3	video	33
document	19	requirements	3	re	30
management	19	testing	3	love	28
quality	19	real	3	apple	28
stakeholders	19	change	3	nice	27
change	18	google	3	week	27
product	18	internet	3	social	25
modeling	17	world	3	read	24
dav93	16	event	3	fun	23
models	16	time	3	home	20
development	15	system	3	rdf	20
environment	15	kataloccounter	3	javafx	18
ka	15	se	3	conference	17
customer	14	post	3	looking	16
functional	14	website	3	site	16
include	14	platform	2	cool	15
particular	14	paypal	2	oracle	13
components	13	people	2	semantic	13
information	13	multitasking	2	html	13
users	13	makes	2	pretty	13
user	12	ldow2010	2	watch	13
domain	11	languages	2	search	13
elicitation	11	mobile	2	news	13
identified	11	industry	2	mobile	13
ieee	11	exactly	2	looks	12
notations	11	edition	2	internet	12
som05	11	drupal	2	service	12
systems	11	discussion	2	flic.kr	11
validation	11	date	2	icio.us	11
project	10	dmdp5b	2	phillyete	11
provide	10	issues	2	odata	11
activities	9	linux	2	oreillymedia	10
conceptual	9	choice	2	linked	10
constraints	9	issues	2	list	10
context	9	languages	2	tonight	9
cost	9	industry	2	sparql	9
customers	9	exactly	2	live	9
cycle	9	mobile	2	oreil.ly	9
life	9	drupal	2	clojure	8

Table D.6: Top 50 terms from classifier, classifier and extended terms in Software Design category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
software	106	linkeddata	105	web	142
design	97	design	82	blog	106
c5	30	web	75	google	83
c6	23	ruby	65	day	74
components	23	data	58	iphone	69
example	23	programming	35	linkeddata	66
bud04	21	rdf	33	ipad	66
structure	19	rails	33	java	51
data	18	code	32	re	46
pre04	18	ia	30	app	42
architecture	16	javascript	30	people	41
bus96	16	github.com	29	talk	36
various	16	user	28	video	33
process	15	based	26	nice	32
abstraction	14	java	25	twitter	32
boo99	14	google	25	social	27
diagrams	14	app	25	love	26
jal97	14	fsharp	24	ux	26
pfl01	14	semantic	22	apple	23
bas98	13	library	22	rdf	21
view	13	nosql	21	read	21
architectural	12	iphone	20	free	20
bos00	12	api	20	fun	20
component	12	linked	19	world	19
describe	12	application	19	javafx	17
languages	12	language	19	search	17
lis01	12	odata	19	thinking	17
quality	12	architecture	19	mobile	16
vs	12	ux	18	conference	15
analysis	11	usability	18	semantic	14
c11	11	rdfa	18	week	14
mar02	11	twitter	17	looking	13
set	11	social	17	cool	13
control	10	sparql	17	service	13
description	10	software	17	internet	12
issues	10	os	16	html	12
methods	10	blog	16	home	12
patterns	10	framework	15	watch	12
requirements	10	information	15	site	12
based	9	day	15	tv	12
c2	9	cool	15	sparql	11
mey97	9	time	15	source	11
notations	9	don	14	news	11
related	9	model	14	odata	11
represent	9	cloud	14	linked	10
bas03	8	php	14	phillyete	10
c9	8	search	14	looks	9
flow	8	awesome	13	future	9
level	8	read	13	pretty	8
measures	8	interaction	13	slides	8
techniques	8	html	13	test	8

Table D.7: Top 50 terms from classifier, classifier and extended terms in Software Construction category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
construction	95	ruby	133	web	139
software	78	java	110	blog	111
testing	40	linkeddata	104	day	85
code	27	web	103	iphone	74
design	22	code	90	google	72
ka	17	data	78	ipad	65
standards	17	rails	71	linkeddata	64
quality	14	javafx	70	app	45
activities	13	programming	69	post	41
mcc04	13	grails	65	nice	35
languages	12	app	65	talk	35
test	12	oracle	62	video	35
activity	10	javascript	59	javafx	30
coding	10	google	57	re	29
topic	10	api	56	love	28
complexity	9	wg21	55	fun	28
detailed	8	cpp	51	twitter	25
integration	8	github.com	46	apple	25
language	8	library	46	read	22
specific	8	jython	44	rdf	21
techniques	8	based	43	free	19
verification	8	rdf	42	home	16
bec99	7	sun	42	news	15
change	7	lua	40	looking	14
example	7	nice	40	cool	13
formal	7	software	38	week	13
include	7	framework	38	looks	12
models	7	language	38	tonight	12
performed	7	python	37	html	12
programming	7	released	36	pretty	12
reuse	7	gt	36	watch	12
unit	7	cool	36	phillyete	12
visual	7	book	36	user	12
align	6	twitter	35	ruby	11
configuration	6	release	35	sparql	10
hun00	6	source	35	live	10
including	6	cloud	34	odata	10
kas	6	development	34	clojure	9
level	6	iphone	33	flic.kr	9
linked	6	plugin	33	oracle	9
management	6	run	33	list	9
notations	6	php	32	reading	9
planning	6	html	31	finally	9
project	6	jquery	31	search	9
ben00	5	beta	31	conference	9
closely	5	check	31	site	9
complex	5	project	31	gt	8
constructing	5	nosql	30	slides	8
engineers	5	apps	30	service	8
ieee	5	developer	29	apps	7
ieee12207-95	5	version	29	internet	7

Table D.8: Top 50 terms from classifier, classifier and extended terms in Software Testing category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
testing	203	testing	161	blog	121
test	167	software	90	google	98
software	110	test	51	ipad	96
techniques	41	bug	30	iphone	85
based	36	tests	23	app	71
faults	34	unit	22	java	68
pfl01	31	tdd	21	book	64
program	31	automation	19	linkeddata	60
bei90	29	qtp	19	ruby	56
process	29	blog	17	nice	54
per95	27	bugs	16	apple	54
different	26	code	15	post	51
reliability	26	web	15	talk	50
tests	25	development	13	grails	47
kan99	20	time	12	video	44
jor02	19	post	12	free	41
ka	18	free	12	love	41
management	18	qa	12	fun	41
system	18	using	12	twitter	38
topic	18	debugging	11	read	34
criteria	17	selenium	11	re	31
quality	17	tester	11	javafx	29
code	16	tools	10	cool	29
set	16	rails	10	watch	29
control	15	refactoring	9	looks	27
evaluation	15	fixed	9	awesome	27
measures	15	agile	9	rails	25
requirements	15	release	8	week	25
failure	14	quality	8	live	23
failures	14	found	8	home	22
functional	14	rspec	8	bad	21
observed	14	talk	8	apps	21
activities	13	suite	8	tonight	21
c9	13	softwaretesting	8	world	21
fault	13	ebook	8	getting	21
input	13	source	8	pretty	20
related	13	help	7	rdf	20
c8	12	bdd	7	social	20
effectiveness	12	book	7	ux	20
product	12	framework	7	thanks	19
behavior	11	re	7	reading	19
defined	11	nice	7	tomorrow	19
information	11	people	7	www.youtube.com	18
objectives	11	soa	7	conference	18
results	11	learn	7	release	17
subarea	11	usability	7	gt	17
c7	10	11g	7	search	17
configuration	10	library	7	mobile	17
lyu96	10	tool	6	news	17
technique	10	testers	6	stuff	15
c1	9	debug	6	site	14

Table D.9: Top 50 terms from classifier, classifier and extended terms in Software Maintenance category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
software	217	refactoring	13	web	136
maintenance	152	rails	12	blog	106
activities	39	windows	12	google	86
process	33	ubuntu	12	iphone	67
development	22	ruby	11	java	65
management	21	linux	11	linkeddata	64
modification	20	app	10	ipad	59
pig97	20	software	10	app	41
planning	20	post	9	twitter	32
change	18	vmware	9	oracle	31
dor02	18	server	9	nice	29
product	18	install	9	love	29
analysis	16	testing	8	apple	28
engineering	16	blog	8	free	26
ieee1219-98	16	apache	8	social	24
maintainer	16	code	8	read	20
pfl01	16	source	7	rdf	20
delivery	15	os	7	javafx	18
maintainability	15	deployment	6	html	15
ieee	14	github.com	6	site	15
processes	14	mac	6	cool	15
impact	13	system	6	semantic	13
iso14764-99	13	running	6	search	13
measures	13	leopard	6	internet	12
request	13	java	6	week	12
configuration	12	nginx	5	watch	11
iec	12	manage	5	phillyete	11
iso	12	oracle	5	news	11
art88	11	kernel	5	odata	11
cost	11	upgrade	5	mobile	11
example	11	installed	5	oreillymedia	10
level	11	project	5	looks	10
quality	11	released	5	live	10
resources	11	moving	4	tonight	10
effort	10	mysql	4	oreil.ly	10
ka	10	engineering	4	sparql	9
life	10	web	4	slides	8
program	10	update	4	clojure	7
testing	10	upgrading	4	apps	7
understanding	10	tools	4	real	7
categories	9	installing	4	programming	6
control	9	based	4	tomorrow	6
issues	9	github	4	top	6
maintainers	9	osgi	4	online	6
measurement	9	tomcat	4	nfjs	5
models	9	easy	3	img.ly	5
tak97	9	dmserver	3	keynote	5
costs	8	module	3	rdfs	5
cycle	8	management	3	semanticweb	5
data	8	dependencies	3	map	5
develop	8	dev	3	ibm	5

Table D.10: Top 50 terms from classifier, classifier and extended terms in Software Configuration Management category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
software	187	git	43	web	128
scm	105	github	28	blog	101
configuration	92	svn	18	google	64
process	65	github.com	13	java	63
change	48	software	8	linkeddata	63
management	44	using	8	iphone	62
items	39	source	7	grails	47
activities	37	subversion	7	ipad	47
control	37	ruby	7	app	37
tools	32	projects	7	people	35
information	28	changes	7	talk	33
item	28	code	5	post	33
project	25	mercurial	4	video	31
changes	24	library	4	apple	31
tool	23	switching	4	twitter	30
buc96	22	support	4	nice	25
baseline	20	engineering	4	free	24
ber92	20	android	4	fun	24
system	20	online	4	oracle	22
development	19	hg	3	javafx	21
release	19	finally	3	rdf	20
example	18	trac	3	re	20
support	18	uwiger	3	world	18
activity	17	development	3	social	18
planning	17	tip	3	read	17
procedures	17	web	3	html	15
product	17	supports	3	looking	15
quality	17	strategy	3	cool	15
cycle	16	server	3	love	15
engineering	16	repository	3	site	14
library	16	project	3	semantic	14
life	16	branch	3	watch	14
status	16	awesome	3	plugin	13
audit	15	built	3	tonight	13
capability	15	apache	3	home	12
reporting	15	mirror	3	looks	11
requirements	15	management	3	pretty	11
various	15	testing	3	news	11
versions	15	top	3	phillyete	11
implementation	14	twitter	3	odata	11
provide	14	knowledge	3	trying	10
authority	13	people	3	flic.kr	9
specific	13	redmine	3	sparql	9
assurance	12	ides	2	linked	9
capabilities	12	index	2	awesome	9
elements	12	html	2	bad	9
measurements	12	hobby	2	getting	9
organizational	12	injoos	2	week	9
scmp	12	hadoop	2	clojure	8
tasks	12	golang	2	internet	8
audits	11	head	2	apps	8

Table D.11: Top 50 terms from classifier, classifier and extended terms in Software Engineering Management category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
software	104	management	23	web	172
management	81	software	22	google	162
project	68	business	14	blog	133
measurement	59	project	12	app	106
process	57	time	9	iphone	100
engineering	45	social	9	ipad	94
example	33	collaboration	8	day	89
requirements	32	web	8	linkeddata	70
organizational	25	post	8	java	65
data	24	agile	7	apple	50
ka	24	data	7	grails	48
processes	21	basecamp	7	nice	47
analysis	18	experience	6	twitter	46
appropriate	18	engineering	6	video	42
procedures	18	enterprise	6	talk	41
quality	18	tracking	6	free	37
activities	17	content	5	love	36
information	15	vs	5	oracle	34
plans	15	tool	5	fun	34
tasks	15	user	5	code	30
tha97	15	source	5	week	30
products	14	startup	5	read	29
risk	14	system	5	re	29
som05	14	release	5	apps	28
rei02	13	developer	4	news	27
resources	13	design	4	javafx	26
undertaken	13	programmers	4	source	26
iso	12	document	4	home	26
methods	12	people	4	search	25
scope	12	oracle	4	cool	24
stakeholders	12	companies	4	rdf	24
adherence	11	code	4	watch	23
c4	11	programming	4	looks	22
evaluation	11	media	4	world	22
organization	11	list	4	mobile	22
pre04	11	available	4	looking	21
relevant	11	architect	4	conference	19
15939-02	10	testing	4	html	18
configuration	10	team	4	semantic	18
criteria	10	help	4	pretty	18
dor02	10	free	4	live	17
effective	10	app	4	tonight	17
managed	10	ibm	4	internet	17
pfl01	10	product	3	list	16
reporting	10	productivity	3	getting	16
review	10	developers	3	service	16
terms	10	development	3	plugin	14
aspects	9	create	3	os	14
based	9	principles	3	ruby	13
c3	9	decisions	3	real	13
change	9	company	3	agile	13

Table D.12: Top 50 terms from classifier, classifier and extended terms in Software Engineering Process category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
process	145	cmmi	31	web	148
software	86	process	15	google	121
processes	51	agile	12	blog	117
measurement	45	engineering	11	day	93
engineering	37	improvement	6	ipad	89
example	36	project	6	iphone	85
change	32	software	6	linkeddata	67
models	30	agility	5	java	66
cycle	26	model	5	app	55
assessment	24	keynote	4	grails	47
life	23	conference	4	twitter	47
organization	23	gp2.10	4	apple	45
implementation	21	scampi	4	nice	44
model	21	useful	4	talk	43
quality	20	training	4	video	42
ieee	19	scrum	4	post	42
management	19	day	4	love	39
improvement	18	processes	4	free	34
outcomes	16	appraisal	4	oracle	32
activities	15	peru	4	fun	32
product	15	book	4	week	30
project	15	methods	3	read	29
iso	13	ml2	3	watch	29
development	12	lima	3	re	28
defined	11	gp2.8	3	home	26
ka	11	team	3	news	24
methods	11	system	3	social	23
tools	11	size	3	world	22
types	11	course	3	search	22
data	10	sepg	3	looking	21
definition	10	review	3	cool	21
analysis	9	satisfy	3	rdf	21
different	9	estimating	3	tonight	20
measures	9	time	3	site	19
method	9	plan	3	conference	19
practices	9	performance	3	javafx	18
related	9	gp2.1	3	html	18
techniques	9	se	3	apps	18
capability	8	using	3	test	17
context	8	ml3	2	release	16
definitions	8	organization	2	tomorrow	15
described	8	makes	2	ruby	15
infrastructure	8	la	2	live	14
meaning	8	panel	2	getting	14
organizational	8	incremental	2	semantic	14
performed	8	include	2	plugin	13
size	8	morning	2	content	13
type	8	jose	2	mobile	13
based	7	guide	2	trying	12
classification	7	gp3.2	2	flic.kr	12
effort	7	gp2.2	2	thanks	12

Table D.13: Top 50 terms of message, classifier and extension in Software Engineering Tools and Methods category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
tools	95	google	60	web	140
software	76	software	58	blog	101
engineering	32	apachecon	57	google	86
methods	26	web	54	iphone	70
dor02	16	apache	54	day	70
management	16	ruby	43	java	65
topic	15	twitter	42	ipad	64
process	14	tool	42	linkeddata	63
pfl01	11	testing	39	app	42
prototyping	11	development	37	using	33
cycle	10	app	37	post	33
life	10	tools	35	people	32
program	9	agile	35	video	32
tool	9	java	34	twitter	31
covers	8	rails	32	talk	28
oriented	8	free	31	re	27
rei96		basecamp	28	apple	26
som05	8	project	24	nice	24
support	8	tinyurl.com	24	javafx	21
test	8	framework	23	grails	21
categories	7	apps	23	fun	21
environments	7	data	23	rdf	20
ka	7	git	23	love	19
pre04	7	released	22	social	19
requirements	7	team	22	world	18
techniques	7	android	22	read	16
behavior	6	process	22	site	15
category	6	oracle	22	search	14
design	6	management	22	html	13
evaluation	6	source	21	looking	13
execution	6	iphone	21	cool	13
integration	6	server	21	week	13
provide	6	windows	20	home	12
specific	6	social	20	flic.kr	11
specification	6	search	19	watch	11
topics	6	vs2010	19	odata	11
assist	5	spring	18	sparql	10
compilers	5	eclipse	18	linked	10
construction	5	asf	18	phillyete	10
data	5	cloud	18	news	10
form	5	api	17	mobile	10
formal	5	list	17	pretty	9
measurement	5	post	17	list	9
modeling	5	available	17	internet	9
processes	5	firefox	17	content	9
product	5	microsoft	17	looks	8
tracking	5	plugin	16	service	8
approaches	4	help	16	trying	7
aspects	4	file	16	tomorrow	7
based	4	github.com	16	bad	7
checking	4	library	15	conference	7

Table D.14: Top 50 terms from classifier, classifier and extended terms in Software Quality category.

Classifier Term	Frequency	Message Term	Frequency	Extension Term	Frequency
software	180	usability	78	web	161
quality	120	ux	60	blog	121
product	57	testing	29	google	117
process	56	user	26	day	99
management	55	software	23	iphone	85
techniques	42	design	19	ipad	83
processes	40	web	19	linkeddata	67
requirements	35	experience	18	java	66
engineering	28	alertbox	15	app	66
sqm	25	ia	14	nice	51
testing	23	qa	14	apple	49
project	21	quality	12	grails	47
activities	19	topic	11	talk	42
v&v	19	news	10	post	42
analysis	18	magazine	10	video	41
development	18	conference	9	twitter	40
products	18	article	9	love	39
reviews	18	research	9	free	36
specific	18	mobile	8	oracle	34
characteristics	17	google	8	re	33
defect	17	ipad	7	fun	32
defects	17	blog	7	home	29
review	17	don	6	week	29
ka	15	useful	6	news	28
sqa	15	code	6	social	25
defined	14	book	6	watch	24
inspection	14	talk	6	looking	22
provide	14	users	6	world	22
purpose	14	online	6	search	22
test	14	w3c	5	rdf	22
include	13	ui	5	tonight	20
maintenance	13	content	5	conference	20
plan	13	interview	5	site	19
plans	13	summit	5	javafx	18
audits	12	sustainable	5	thanks	18
ensure	12	softwaretesting	5	cool	18
organization	12	assurance	5	looks	17
system	12	engineering	5	apps	16
cost	11	facebook	5	getting	16
customer	11	mex	4	live	15
failure	11	development	4	html	15
improvement	11	job	4	internet	15
information	11	tips	4	content	15
planning	11	interfaces	4	tomorrow	14
related	11	standards	4	release	14
standard	11	html	4	awesome	14
technical	11	information	4	ruby	14
example	10	search	4	mobile	14
models	10	review	4	plugin	13
verification	10	site	4	semantic	13
activity	9	social	4	bad	13

APPENDIX E

RETRIEVAL EVALUATION EXPERIMENT RESULTS

E.1 Retrieval Evaluated Score

Table E.1: WPR@5, WPR@10 and WPR@20 of baseline treatment RT_0 and social context treatment RT_1

keyword	Baseline treatment RT_0			Social context treatment RT_1		
	WPR@5	WPR@10	WPR20	WPR@5	WPR@10	WPR@20
tool	0	0.163	0.395	0.6	0.436	0.528
development	0.666	0.618	0.4	0.8	0.745	0.738
code	0	0.09	0.261	0.466	0.363	0.428
ux	0.866	0.654	0.576	0.733	0.581	0.59
agile	1	0.8	0.69	1	0.89	0.747
search	0.533	0.69	0.761	1	0.836	0.852
application	0.2	0.272	0.438	0.533	0.618	0.614
usability	0.4	0.654	0.814	0.466	0.672	0.819
project	0.466	0.381	0.433	0.6	0.654	0.58
system	0.4	0.4	0.457	0.533	0.636	0.595
java	0.333	0.509	0.423	0.333	0.381	0.428
management	0.333	0.472	0.528	0.333	0.4	0.538
css	1	1	1	1	1	1
process	0.733	0.69	0.504	0.533	0.563	0.566
window	0.333	0.272	0.166	0.333	0.272	0.19
ixda	1	1	1	1	1	1
file	0.533	0.509	0.485	0.8	0.69	0.652
tester	1	1	0.866	1	1	0.866
jquery	0.4	0.545	0.609	0.866	0.636	0.652
qtp	0.733	0.836	0.857	0.933	0.89	0.871
source	1	0.909	0.923	1	0.945	0.938
app	0.066	0.236	0.28	0.066	0.272	0.347
iphone	0.4	0.29	0.385	0.333	0.345	0.409
programming	0.466	0.672	0.785	0.666	0.781	0.8
interaction	0.8	0.854	0.895	0.866	0.872	0.88
service	0	0.109	0.271	0.6	0.49	0.385
semanticweb	1	1	1	1	1	1
designer	0.066	0.381	0.557	0.572	0.454	0.59
javascript	0.066	0.345	0.547	0.733	0.618	0.647
mobile	0.333	0.454	0.623	0.533	0.6	0.7
ui	0.133	0.272	0.48	0.533	0.454	0.571
quality	0	0.145	0.295	0.133	0.236	0.28
flash	1	1	0.961	1	1	0.99
html	0.6	0.545	0.557	0.6	0.454	0.461
developer	0.666	0.818	0.78	0.8	0.854	0.88
architecture	0.933	0.89	0.923	1	0.963	0.942
win7	0.432	0.345	0.48	0.333	0.327	0.48
bug	0.666	0.818	0.904	1	0.981	0.947
client	0	0.163	0.157	0.533	0.327	0.3
interface	0.333	0.436	0.571	0.866	0.781	0.728
microsoft	0.333	0.545	0.69	0.866	0.818	0.809
database	0.4	0.6	0.585	0.933	0.818	0.795
automation	1	1	1	1	1	1
scala	0.066	0.218	0.395	0.066	0.29	0.414
rdf	1	0.981	0.947	1	1	1
framework	0.666	0.818	0.9	0.733	0.836	0.909
library	0.333	0.436	0.504	0.666	0.654	0.695
security	0.266	0.436	0.471	1	0.854	0.757
javafx	0	0.072	0.157	0	0.018	0.142
plugin	0.533	0.509	0.557	0.933	0.672	0.571
Average	0.48972	0.55704	0.60486	0.68452	0.65954	0.67242

Table E.2: DCG@5, DCG@10 and DCG@20 of baseline treatment RT_0 and social context treatment RT_1.

keyword	Baseline treatment RT_0			Social context treatment RT_1		
	DCG@5	DCG@10	DCG@20	DCG@5	DCG@10	DCG@20
tool	2.9485	5.4934	9.7693	4.5794	6.7969	11.3102
development	4.9662	7.2509	9.7476	5.0794	7.9805	12.4723
code	2.9485	4.8998	9.1475	4.0794	6.2646	10.0322
ux	5.4662	7.6514	11.5999	5.2660	7.1502	11.6376
agile	5.8969	8.0965	12.0798	5.8969	8.1816	12.4269
search	4.4662	7.6564	12.1439	5.8969	8.3976	13.1596
application	3.4485	5.7008	10.4316	4.4662	7.0519	11.0633
usability	4.2660	7.4562	12.4496	4.3969	7.5871	12.5805
project	4.3791	6.3076	10.7824	4.8353	7.4354	11.1748
system	4.3353	6.5648	10.3493	4.7660	7.6407	11.1186
java	3.9662	6.8554	9.5798	3.9662	6.4669	9.9503
management	3.8791	6.4648	10.6899	3.9662	6.4848	10.7470
css	5.8969	9.0871	14.0805	5.8969	9.0871	14.0805
process	5.2660	7.5183	10.4951	4.7660	7.3074	11.0479
window	3.9662	5.5613	8.0581	3.9485	5.8998	8.6411
ixda	5.8969	9.0871	14.0805	5.8969	9.0871	14.0805
file	4.4662	6.4175	10.6383	5.3969	7.6373	11.6657
tester	5.8969	9.0871	10.1855	5.8969	9.0871	10.1855
jquery	4.3353	7.2364	11.6950	5.4662	7.3624	11.8967
qtp	5.2660	8.4562	12.7153	5.5101	8.7003	12.9594
source	5.8969	8.7309	13.4967	5.8969	8.7717	13.7651
app	3.3353	5.5792	9.0450	3.3353	5.6200	9.5891
iphone	4.0101	5.6052	9.8855	3.9485	6.4934	10.4473
programming	4.3969	7.5871	12.3246	4.8969	7.7861	12.5006
interaction	5.3969	8.5871	13.3407	5.4662	8.6564	13.3872
service	2.9485	5.1779	8.9395	4.5101	6.4206	9.6594
semanticweb	5.8969	9.0871	14.0805	5.8969	9.0871	14.0805
designer	3.3353	6.5255	10.7589	3.8791	6.7131	10.9603
javascript	3.3353	6.2245	10.5113	5.2660	7.4512	11.7066
mobile	3.8791	6.7131	11.4363	4.4485	7.6387	12.1374
ui	3.3791	5.9121	10.6353	4.4485	6.6924	11.6858
quality	2.9485	5.4814	9.2119	3.3791	5.6086	9.0690
flash	5.8969	9.0871	13.5992	5.8969	9.0871	13.8492
html	4.8353	7.0757	10.8756	4.5101	6.3942	10.6048
developer	4.8969	8.0871	12.0645	5.3969	8.5871	13.1045
architecture	4.8791	8.0693	13.0628	4.8969	7.7861	12.7795
win7	3.9485	6.5043	10.5473	3.9485	6.4899	10.5388
bug	4.8969	8.0871	13.0805	5.8969	8.7981	13.7915
client	2.9485	5.2331	7.9696	4.4485	6.0436	9.7742
interface	3.9662	6.5112	11.2344	5.4662	8.0399	12.5148
microsoft	3.9662	7.1564	11.8709	5.4662	8.3410	13.0555
database	4.2660	6.8661	10.6337	5.5101	8.3669	12.8544
automation	5.8969	9.0871	14.0805	5.8969	9.0871	14.0805
scala	3.3353	5.8360	9.5976	3.3353	5.9031	9.6648
rdf	5.8969	8.7981	13.7915	5.8969	9.0871	14.0805
framework	4.8969	8.0871	12.8529	5.2660	8.4562	13.4496
library	3.9662	6.2509	10.7343	4.9662	7.8231	11.8518
security	3.8353	6.6693	10.1249	5.8969	8.4154	12.6199
javafx	2.9485	5.1481	8.3538	2.9485	4.8326	8.5297
plugin	4.7660	6.7173	11.4607	5.5101	7.6953	11.6335
Average	4.4152	7.0666	11.2064	4.9300	7.5556	11.7999

Biography

Warut Surpat was born on November 15, 1984, in Bangkok, Thailand. He received a Bachelor's degree of computer science from Faculty of Science, Chulalongkorn University in March 2006. After that, he worked as a software developer at Sony Device Thailand. Simultaneously, he is a senior developer at Wone'. His life in Sony and Wone' was continued for one and a half year. Mainly works of him are user interface and web standard engineering. Before deciding to take further study, he joined Accenture for a few months as an user interface developer and web standard engineer. In 2008, he began his Master degree in Software Engineering at Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University. His field of interest includes various research topics such as Social Network Analysis, Information Retrieval, Software Design, Human Computer Interaction, Artificial Intelligent, Machine Learning and Data Visualization.

List of Publication

1. Warut Surapat, Nakornthip Prompoon, "Personalized Software Engineering Related Message Storage and Classification over Microblogging Application", Proceedings of the 7^{th} International Joint Conference on Computer Science and Software Engineering (JCSSE 2010), Bangkok, Thailand, May 12-14, 2010.
2. Warut Surapat, Nakornthip Prompoon, "Social Clues Powered, Personalized Software Engineering Messages Classification", 10^{th} International Symposium on Communications and Information Technologies 2010 (ISCIT 2010), Tokyo, Japan, October 26-29, 2010.