Content Quality in Web 2.0 Services - Analysis, Detection, Systems and Enhancement

by

Denzil Correa

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science & Engineering

at

Indraprastha Institute of Information Technology, Delhi

Committee in charge:

Prof. Ashish Sureka, Advisor
Prof. Premkumar Devanbu (University of California, Davis)
Prof. Vasudev Varma (IIIT-Hyderabad)
Dr. Srinivas Padmanabhuni (Principal Research Scientist, Infosys Labs)

December 2014

Abstract

There has been a proliferation of Web 2.0 sites on the Internet. Contemporary Web 2.0 sites like Facebook and Twitter are primarily driven by *user generated content* (UGC). On the other hand, the volume of content by such user contributions has been increasing rapidly. However, user generated content may not conform to the set of guidelines and rules of the websites. Subpar content can severely affect user engagement, retention and also have an adverse impact on information retrieval systems. Therefore, there is an impending need to manage and enhance content quality on Web 2.0 sites. In this thesis, we investigate three broad objectives – (1) Low quality content, (2) Content Quality Systems and (3) Information Retrieval Enhancement. In order to address these objectives, first - we look at low quality questions on a popular programming based community based question answering (CQA) website called Stackoverflow. We analyze user behavior, content patterns and also build supervised machine learning based predictive systems to detect low quality questions. In context to the second objective, we look at enhancing content quality on Issue Tracking Systems - a popular artifact used by developers during the software maintenance lifecycle. We conduct surveys from software practitioners to understand the needs of the community and discover that developers frequently use the Internet for their daily tasks. In order to reduce the context switch for software maintenance professionals, we develop two systems – (i) CQA integration with Issue Tracking Systems and (ii) Web Reference Management Browser Plugin. We develop both these systems to reduce cognitive load on software maintenance professionals during their daily tasks. In context to the third objective, we look at quality enhancement on social media to help information retrieval systems. Concretely, we propose a new algorithm to utilize social interactions to discover homogeneous topic-based communities on a social network. To address the challenge of scalability, our algorithm only visits required portions of the network based on an expectation-maximization approach. Further, we also propose an algorithm for tag recommendation on social media. Specifically, we utilize Twitter to suggest tags to external linked media like Flickr, Youtube and Soundcloud. In conclusion, we look at different perspectives for quality analysis, systems, detection and enhancement of content on Web 2.0 sites.

I dedicate this thesis to my father Rosario Franco Correa who is not here to live this day. He would not agree with me but would always defend the right to speak my alternate and extremely crazy view points. I miss him.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank mom and sister for bearing me throughout my PhD. I tested their patience levels but they never gave up on me , even for a second. I would also like to thank my grandmother (*Mai*) who despite her age ensured that her love was always a driving force. Thank You *Mai*! I can not help but feel humbled by the gratitude of many family members, friends and neighbors. Many of them could not understand what I did or the significance of my research but they never doubted my ability.

A PhD is incomplete without an advisor. I was lucky to have Ashish as my advisor and professional parent who stood with me through thick and thin, literally. Ashish provided me a near perfect platform to demonstrate my potential and encouraged me at every step.

There were also many professional colleagues whom I'd like to acknowledge. Anupama for the endless and passionate research discussions, creative and colorful block diagrams, formal equations and paper draft reviews. David Lo for giving me an internship opportunity at Singapore Management University(SMU). Professor Ee-Peng Lim who guided me during my SMU internship and honed my critical thinking skills. Fabricio Benevenuto for helping me during the latter stages of my PhD with long talks which cleared up my thoughts for my life after PhD. Krishna Gummadi for giving me an opportunity to work with him at Max Planck Institute for Software Systems. Finally, Professor Pankaj Jalote who decided to take the risk to accept me as a graduate student at IIITD.

I stand here with a PhD on the shoulder of these aforementioned giants. This PhD is theirs, as much it is mine.

Thesis Publications

1. **Denzil Correa** and Ashish Sureka. (2014, April). Chaff from the wheat: characterization and modeling of deleted questions on stack overflow. In Proceedings of the 23rd international conference on World wide web (pp. 631-642). International World Wide Web Conferences Steering Committee.

2. **Denzil Correa**, Sangeeta Lal , Apoorva Saini and Ashish Sureka. (2013, December). Samekana: A Browser Extension for Including Relevant Web Links in Issue Tracking System Discussion Forum. In Software Engineering Conference (APSEC, 2013 20th Asia-Pacific (Vol. 1, pp. 25-33). IEEE.

3. **Denzil Correa** and Ashish Sureka. (2013, October). Fit or unfit: analysis and prediction of'closed questions' on stack overflow. In Proceedings of the first ACM conference on Online social networks (pp. 201-212). ACM.

4. **Denzil Correa** and Ashish Sureka. (2013, June). Integrating issue tracking systems with community-based question and answering websites. In Software Engineering Conference (ASWEC), 2013 22nd Australian (pp. 88-96). IEEE.

5. **Denzil Correa**, Ashish Sureka and Mayank Pundir. (2012, November). iTop: interaction based topic centric community discovery on twitter. In Proceedings of the 5th Ph. D. workshop on Information and knowledge (pp. 51-58). ACM.

6. **Denzil Correa** and Ashish Sureka. (2011, October). Mining tweets for tag recommendation on social media. In Proceedings of the 3rd international workshop on Search and mining user-generated contents (pp. 69-76). ACM. Chicago

Chapter 1

Introduction

The meteoric rise of the Internet has led to an increase of Web 2.0 websites on the World Wide Web (WWW). Web 2.0 websites are a paradigm shift from the traditional websites and are interactive in nature allowing user contributions. Examples of Web 2.0 websites include social media like Facebook, Twitter, Youtube, Foursquare and Flickr [4, 5, 14, 10, 3]; Community Based Question-Answering websites like Stack Overflow, Yahoo! Answers and Quora [7, 13, 9]; Software Engineering Issue Tracking Systems - Chromium and Android [1, 2]. Most of the aforementioned contemporary Web 2.0 websites are primarily driven by user-generated content (UGC). Over the last few years, there has been an explosion of UGC on the WWW owing to an increase in Web 2.0 websites. For example, Twitter – a very popular microblogging website – attracted more than 500M tweets per day as of October 3, 2013 [24] while Stack Overflow – a popular programming based Question-Answering website – has 7.1M questions and 12M answers [8]. All thriving Web 2.0 sites have shown an increasing trend of UGC volume and forecasts predict that the future trends will show a more sharp increase. Such rapid increases in UGC volumes has led to many interesting challenges and opportunities for researchers across the globe. In this thesis, we focus on an important aspect of UGC on the Internet - *content quality* viz. quality of the UGC on the WWW.

1.1 Content Quality - Motivation

The traditional model of publishing (Web 1.0) was a *publish* only model in which website owners published information for Internet users to consume. The Web 2.0 publishing model is a two-way street which includes the Internet users publishing content on the website. In fact, most Web 2.0 sites thrive mostly on UGC while the only contributions from the website owners are content publishing platforms or provisions. Similar to content on traditional WWW, UGC on Web 2.0 websites can be rich sources of information. Studies have shown that Twitter provides quality temporally relevant information like news and breaking events [83, 124] and Stack Overflow is now the primary website for programming related information [93]. In few geographical niche markets, these avenues of information are considered more significant than the traditional web [16]. Despite their popularity and usefulness, these websites have a very low publication barrier and are

primarily driven by users of these sites. Thus, unlike the traditional WWW - where there are intermediate health checks on the content - in Web 2.0 there is a large variance in the quality of UGC on such websites. Despite the variation in quality, websites like Twitter and Stack Overflow are used by many users around the world to search and discover information for niche needs and purposes [83, 93]. In many practical use cases, these websites provide quicker and more relevant, accurate results for certain types of information requirements.

The voluminous UGC available on such Web 2.0 sites provides interesting opportunities and challenges for research. There is little or no intermediate health checks of content the quality of content is remarkably varied than traditional sources [37]. The high variance in information quality distribution provides unexpected and complex challenges for current systems. Since Web 2.0 websites are free to user and a low publication barrier, the maintenance of content quality on the website is a challenging task. However, maintenance of content quality is a challenge due to the sheer volume of content and the number of users. Popular Web 2.0 websites are repeatedly targeted by malicious users for various egregious purposes like spam [71, 85] and phishing [49, 26]. In addition, users of the website may also post content which is against the rules of the website and therefore, can also be construed as low quality given the context. Low quality content has a debilitating effect on information retrieval systems and also effects user experience [27, 93]. A manual process to sift and quarantine content on the site is practically infeasible due to the sheer volume of users and content on such Web 2.0 services. Most of these websites rely on crowd sourced mechanisms like community moderators who quarantine content to maintain the hygiene on the website. However, inappropriate or belligerent moderation can have undesirable effect on the users [61]. In addition, to moderation of low quality content it is also essential to provide an encouraging platform and interface for users to maintain content quality. It is important to identify key indicators of content quality and help users to indulge in these indicators. All these factors make the maintenance of content quality on Web 2.0 sites a challenging and contemporary research agenda. In this thesis, we investigate different challenges about content quality in three diverse and different contemporary Web 2.0 services – Stack Overflow (a popular programming based question-answering website), Issue Tracking Systems (an integral part of software maintenance process) and Twitter (a popular microblogging platform). In the next sections, we explain the broad objectives of this thesis.

1.2 Objectives of Thesis

Content quality on Web 2.0 services is a important and major challenge. Therefore, it is important to look at the different aspects that affect content quality on WWW. In this thesis, we look to make progress on three aspects of content quality – (1) Low Quality Content, (2) Content Quality Systems and (3) Information Retrieval. We now explain each of these three aspects.

Low Quality Content

Web 2.0 services exist and thrive on user generated content. Therefore, there is a high degree of variance in the quality of the content due to minimal publishing barriers. The content could be extremely low in quality like spam, off-topic and abuse. Such content may adversely affect the user experience and also severely impact Information Retrieval algorithms. Hence, it is important to study the factors affecting low quality content. Some research questions include – *Why do users post low quality content?*, *What are the content characteristics of low quality content?* and *What is the nature of low quality content?*. Our objective is to understand the phenomena of low quality content from the user perspective. In addition, most Web 2.0 services employ moderators to manually curate and weed out low quality content. The answers to our previous questions can also help us build systems to automatically detect low quality content and thus aid the moderators in their daily tasks.

Content Quality Systems

Since the primary function of Web 2.0 services is to aid users to contribute content; the user becomes the focal point of the content quality ecosystem. Therefore, it is imperative that the user is provided resources to help her in content contribution. In the previous objective, we expect to discover content characteristics of low quality content. In this part of the thesis, our objective is to build on these content characteristics to develop systems for content quality enhancement. The system(s) should directly contribute to the daily work flow of the user and have a positive impact on her productivity. Overall, these systems should enhance the user experience on the respective Web 2.0 service and encourage users to contribute better quality content to these services.

Information Retrieval Enhancement

The user generated content on Web 2.0 services may have missing information. Missing information has an adverse impact on Information Retrieval systems and in turn the user experience. Therefore, it is important to add relevant contextual information to content to aid Information Retrieval systems. In this thesis, we look at solutions to improve the content quality by adding relevant metadata to user generated content. In addition to missing information, the search and retrieval algorithms should leverage high quality content to discover accurate and meaningful results to the end user. Hence, we aim to build information retrieval algorithms to leverage quality content for Information Retrieval enhancement.

1.3 Contributions of Thesis

Analysis and Detection of Low Quality Content

One of the main objectives of this thesis is to study low quality content. Here, we aim to detect low quality content on user generated content on Community Based Question Answering

(CQA) websites. Concretely, we investigate quality of questions on Stack Overflow which is a very popular CQA website for programmers. Stack Overflow is a free, open website to all users and therefore, maintenance of content quality on such a large scale social collaborative platform is a challenge [27]. Stack Overflow has detailed, explicit guidelines on posting questions and it maintains a firm emphasis on following a question-answer format. The community strongly discourages questions which could generate chit-chat, opinions, polls etc. and employs elected moderators to ensure content quality maintenance. Opinion-based questions and questions which have a tendency to generate discussions rather than answers are categorically considered inappropriate. Some examples of such questions include (but not limited to) homework questions, product or service recommendations, non-programming related and polls. Stack Overflow is driven by the goal to be an exhaustive knowledge base on programming related topics and hence, the community would like to ensure minimal possible *noise* on the website. However, despite of the presence of question posting guidelines and an ebullient moderation community, a significant percentage of questions on Stack Overflow are extremely poor in nature. Questions are a fundamental aspect of any CQA website and the presence of poor quality content may affect user experience. Question quality control also plays a significant role in a CQA's functioning and popularity. Prior work also shows that poor quality content on a CQA website may drive users away from the platform and adversely affect the traffic [93]. Therefore, it is important to study poor quality questions and develop mechanisms to minimize them on the website.

Closed Questions

Questions posted on Stack Overflow which are not related to programming topics, are marked as 'closed' by experienced users and community moderators. A question can be 'closed' for five reasons – *duplicate, off-topic, subjective, not a real question* and *too localized*. In this work, we present the first study of 'closed' questions on Stack Overflow. We download 4 years of publicly available data which contains 3.4 Million questions. We first analyze and characterize the complete set of 0.1 Million 'closed' questions. Next, we use a machine learning framework and build a predictive model to identify a 'closed' question at the time of question creation.

One of our key findings is that despite being marked as 'closed', *subjective* questions contain high information value and are very popular with the users. We observe an increasing trend in the percentage of closed questions over time and find that this increase is positively correlated to the number of newly registered users. In addition, we also see a decrease in community participation to mark a 'closed' question which has led to an increase in moderation job time. We also find that questions closed with the *Duplicate* and *Off Topic* labels are relatively more prone to reputation gaming. Our analysis suggests broader implications for content quality maintenance on CQA websites. For the 'closed' question prediction task, we make use of multiple genres of feature sets based on - user profile , community process, textual style and question content. We use a state-of-art machine learning classifier based on an ensemble framework and achieve an overall accuracy of 70.3%. Analysis of the feature space reveals that 'closed' questions are relatively less informative and descriptive than non-'closed' questions. To the best of our knowledge, this is the first experimental study to analyze and predict 'closed' questions on Stack Overflow.

Deleted Questions

Stack Overflow has explicit, detailed guidelines on *how to post* questions and an ebullient moderation community. Despite these precise communications and safeguards, questions posted on Stack Overflow can be extremely off topic or very poor in quality. Such questions can be *deleted* from Stack Overflow at the discretion of experienced community members and moderators. We present the first study of *deleted questions* on Stack Overflow. We divide our study into two parts – (i) Characterization of *deleted questions* over $\approx$ 5 years (2008-2013) of data, (ii) Prediction of *deletion* at the time of question creation.

Our characterization study reveals multiple insights on question deletion phenomena. We find that it takes substantial time to vote a question to be deleted but once voted, the community takes swift action. We also see that question authors delete their questions to salvage reputation points. We notice some instances of accidental deletion of good quality questions but such questions are voted back to be undeleted quickly. We discover a pyramidal structure of question quality on Stack Overflow and find that *deleted questions* lie at the bottom (lowest quality) of the pyramid. We also build a predictive model to detect the *deletion* of question at the creation time. We experiment with 47 features – based on *User Profile*, *Community Generated*, *Question Content* and *Syntactic* style – and report an accuracy of 66%. Our findings reveal important suggestions for content quality maintenance on community based question answering websites. To the best of our knowledge, this is the first large scale study on poor quality (*deleted*) questions on Stack Overflow.

Content Quality Systems

In the second part of our thesis, our objective is to build systems to improve content quality. Here, we focus on building systems for software maintenance professionals. We draw from insights from our study on low quality and build systems to address some of the indicators of content quality. Research shows that software developers spend a significant amount of their time outside their development environment (such as Eclipse Integrated Development Environment) and inside their Web Browser searching and navigating information available on the Web required for problem-solving [69]. Prior research reveals that programmers spent 19% of their programming time on the Web to accomplish several different kinds of activities (such as learning of unfamiliar concepts, language syntax, API or function usage, reading tutorials or how-to articles, connecting high-level knowledge to implementation details) [45]. Communication, collaboration, exchanging knowledge and searching for information is not only a common activity during development but also during bug resolution and bug fixing [40, 140]. Web searching, browsing and navigation, referring to online resources and posting web-links to resources is not only done during programming but also during issue resolution process. Previous research shows that developers spent considerable amount of time outside their development environment and in their web-browser resulting in a context-switch between the development environment and browser [64, 57, 54, 35, 51, 44]. Therefore, tools and systems to integrate web resources into issue tracking systems can be extremely useful for software maintenance professionals.

Integrating Issue Tracking Systems with Community Based Question-Answering Websites

Issue tracking systems such as Bugzilla are tools to facilitate collaboration between software maintenance professionals. Popular issue tracking systems consists of discussion forums to facilitate bug reporting and comment posting. We observe that several comments posted in issue tracking system contains link to external websites such as YouTube (video sharing website), Twitter (micro-blogging website), Stackoverflow (a community-based question and answering website for programmers), Wikipedia and focused discussions forums. Stackoverflow is a popular community-based question and answering website for programmers and is widely used by software engineers as it contains answers to millions of questions (an extensive knowledge resource) posted by programmers on diverse topics. We conduct a series of experiments on open-source Google Chromium and Android issue tracker data (publicly available real-world dataset) to understand the role and impact of Stackoverflow in issue resolution. Our experimental results show evidences of several references to Stackoverflow in threaded discussions and demonstrate correlation between a lower mean time to repair (in one dataset) with presence of Stackoverflow links. We also observe that the average number of comments posted in response to bug reports are less when Stackoverflow links are presented in contrast to bug reports not containing Stackoverflow references. We conduct experiments based on textual similarly analysis (content-based linguistic features) and contextual data analysis (exploited metadata such as tags associated to a Stackoverflow question) to recommend Stackoverflow questions for an incoming bug report. We perform empirical analysis to measure the effectiveness of the proposed method on a dataset containing ground-truth and present our insights. We present the result of a survey (of Google Chromium Developers) that we conducted to understand practitioner's perspective and experience.

Samekana: A Browser Extension for Including Relevant Web Links in Issue Tracking Systems

Several widely used Issue tracking systems (such as Google Issue Tracker and Bugzilla) contains an integrated threaded discussion forum to facilitate discussion between the development and maintenance team (bug reporters, bug triagers, bug fixers and quality assurance managers). We observe that several comments (and even bug report descriptions) posted to issue tracing system contains links to external websites as references to knowledge sources relevant to the discussion. We conduct a survey (and present the results of the survey) of Google Chromium Developers on the importance and usefulness of web references in issue tracking system comments and the need of a web-browser extension which facilitates easy organization and inclusion of web-links in the post. We conduct a characterization study on an experimental dataset from Google Chromium Issue Tracking system and present results on the distribution of number of links in the dataset, categorization of links into pre-defined classes (such as blogs, community based Q&A websites, developer discussion forums, version control system), correlation of number and types of links with various bug report types (such as security, crash, regression and clean-up) and relation between presence of links and bug resolution time. Survey results and data characterization study motivate the need of building a developer productivity tool to facilitate web-link (as references)

organization and inclusion in issue tracking system comments. We present a Google Chromium Web Browser Extension called as *Samekana* and publish the extension on Google Chromium Web Store[1] which can be freely downloaded by users worldwide. The extension contains features such as annotating (using tags, title and description) and saving web references pertaining to multiple bug reports and tasks and then posting it as bibliography (for easy citation and reference) in issue tracking system comments.

Information Retrieval enhancement

Web 2.0 services provide a platform to subscribers to publish content. Most of these websites are open to all Internet users and encourage users to contribute content in various forms. Prior research reveals issues with the quality of user generated content [80, 89]. Therefore, it is important to look for solutions which *enhance* content quality on Web 2.0 sites. We look into the problem of tag recommendation to enhance metadata for multimedia content. Web 2.0 sites also allow users to *interact* with each other using various mechanisms. These interactions lead to formation of virtual and implicit communities around specific topics. Mining interactions can be a potentially useful quality enabler to extract implicit topic-centric communities. Hence, we look at algorithms to discover interaction-based topic-specific communities on social media.

Mining Tweets for Tag Recommendation on Social Media

Automatic tag recommendation or annotation can help in improving the efficiency of text-based information retrieval on online social media services like Blogger, Last.FM, Flickr and YouTube. We investigate alternate solutions for tag recommendations by employing a *Wisdom of Crowd* approach in a mashup framework. In particular, we mine tweets on Twitter and use their hashtag(s) and content to annotate videos on Flickr, Photobucket, YouTube, Dailymotion and SoundCloud. We crawl Twitter to collect a random sample of tweets containing Flickr, Photobucket, YouTube, Dailymotion and SoundCloud URLs. We then recommend tags for these services using hashtag(s) and content present in tweets. We use a hybrid technique (automated and manual) to validate our results on different subsets (presence / absence of hashtags, presence / absence of media tags) of data. Experimental results demonstrate that the proposed solution approach is effective and reliable.

Interaction Based Topic Centric Community Discovery on Twitter

Automatic detection of communities (or cohesive groups of actors in social network) in online social media platforms based on user interests and interaction is a problem that has recently attracted a lot of research attention. Mining user interactions on Twitter to discover such communities is a technically challenging information retrieval task. We present an algorithm – *iTop* – to discover interaction based topic centric communities by mining user interaction signals (such as @-messages and retweets) which indicate cohesion. *iTop* takes any topic as an input keyword and exploits local

[1] `https://chrome.google.com/webstore/detail/samekana/mjhpnkbpjcfijhfiblkfaeiiooineblf`

information to infer global topic-centric communities. We evaluate the discovered communities along three dimensions: graph based (node-edge quality), empirical-based (Twitter lists) and semantic based (frequent n-grams in tweets). We conduct experiments on a publicly available scrape of Twitter provided by InfoChimps via a web service. We perform a case study on two diverse topics - 'Computer Aided Design (CAD)' and 'Kashmir' to demonstrate the efficacy of *iTop*. Empirical results from both case studies show that *iTop* is successfully able to discover topic-centric, interaction based communities on Twitter.

1.4 Organization of Thesis

The rest of this thesis is organized as follows. Chapter 2 presents literature survey in context to the objectives of our thesis. Chatper 3 and Chapter 4 propose solution approach to analyze and detect low quality content on Stackoverflow CQA. We then move to build systems to enable quality in technologies used during software maintenance tasks. Concretely, Chapter 5 proposes a solution to integrate CQA websites with Issue Tracking Systems to assist software maintenance professionals and bug reporters. In Chapter 6, we analyze link sharing patterns in Issue Tracking Systems and build a browser plugin to help software maintenance professionals manage their web references. We then move on to the inclusion of quality indicators for effective information retrieval on social media. Chapter 7 proposes to utilize high quality interactions amongst users in social media to discover implicit homogeneous topic-based communities. In Chapter 8, we look into leveraging Twitter to recommend tags on multimedia posts to improve information retrieval. In the end, we give a brief summary of our thesis in Chapter 9.

Chapter 2

Literature Survey

In this chapter we present a literature survey of prior work in context to the thesis. Earlier, we stated that the broad objectives of this thesis is – analysis, detection of low quality content and building quality enablers for search in UGC media like issue tracking systems and social media. In recent past, there has been a lot of work on quality of content on in these domains. We perform a review of literature with respect to the objectives of our thesis.

2.1 Background

We give a brief background of the Web 2.0 services we perform our study.

Community Based Q&A sites – Stack Overflow

Community driven Question Answering (CQA) websites like Stack Overflow, Quora and Yahoo! Answers are popular contemporary genre of websites on the Internet. CQA websites follow a standard Q&A format where a user asks a question on a problem she faces; while other users (who may have some prior expertise) respond with their answers on the question. Effectively, CQA websites follow a crowd sourced model in which the knowledge of experts is exploited to form a large scale knowledge base on variety of topics. Stack Exchange is a platform which provides libraries to deploy topic-based community powered Q&A websites [75]. Stack Exchange is a growing network of thematic question-answering websites with each website dedicated to a specific field of expertise [75]. It consists of 107 CQA websites with 4.1M users, 7.3M questions, 13.2 M answers and 8.5M visits per day.[1] CQA websites on Stack Exchange span across different orthogonal themes like *Technology* (Web Applications, Game Development), *Culture* (Travel, Christianity), *Arts* (Photograph, Scientific Fiction) and *Sciences* (Mathematics, Physics). Stack Overflow is a programming based CQA and the most popular Stack Exchange website consisting of 7.1M questions, 12M answers and 2M+ registered users on its website [9].

[1] `https://stackexchange.com/about`

Stack Overflow is the first and most popular Stack Exchange website which caters to the benefits of professional programmers and programming enthusiasts. It is a free and open Q&A website where users can ask programming related questions. Stack Overflow maintains a strong emphasis on question-answer based format of the site and strongly discourages discussion or *chit-chat*. The community strongly discourages questions which could generate chit-chat, opinions, polls etc. and employs elected moderators to ensure content quality maintenance. Stack Overflow also encourages question askers to post details about the problem they face and recommend that questions contain source code. In addition, it is encouraged to mention details about previously tried but unsuccessful solutions along with the actual question.

In particular, questions on the topics which contain specific programming problems, software algorithms, coding techniques and software development tools are recommended and considered fit for its Q&A format. An intricate community based voting process is followed to reward users for good quality questions and answers. Relevant, technically challenging and good question-answers are rewarded by the community with *votes*. Similarly, answers which address the problem encountered by the original question can be voted *accepted*. This voting process allows post owners to earn a *reputation* which is a reflection of their contribution worth to the Stack Overflow community. Conversely, the same voting process can lead to penalties on the post owner's *reputation* due to low quality posts like wrong answers, spam and advertisements. *Badges* (the online equivalent of medals) are awarded to users as incentives to highlight special achievements based on community participation. This community based *reputation reward* process helps to ensure a reasonable degree of high quality content on the website and weed out low quality content. Stack Overflow is a free, open website to all users and therefore, maintenance of content quality on such a large scale social collaborative platform is a challenge [27]. In this thesis, we specifically look into the aspect of question quality on Stack Overflow.

Issue Tracking Systems

Issue Tracking Systems (commonly referred to as Bug or Defect Tracking Systems) such as Bugzilla, Google Issue Tracker, JIRA and Mantis provides a platform to support software maintenance activities such as issue reporting, tracking and resolution. Issue (such as a bug report or a feature enhancement request) tracking and resolution is a social and collaborative process in which an issue tracker serves as a communication hub and channel between the users, developers and QA (Quality Assurance) team [40]. Also, software maintenance process can consume more than 50% of the total life cycle costs [129]. Therefore, Issue Tracking Systems are an important cog in the wheel of the software maintenance process.

A typical process on an issue tracking systems requires the bug reporter to report a bug. The bug fixers may comment on the bug to seek further clarifications from the reporter or use it as a communication medium to discuss bug resolution strategies and road-maps. There could be many other features in an issue tracking system but textual description of bug reports and comments remain the most significant aspects of the process. There are detailed guidelines on bug reporting

however, reporters and fixers may choose to ignore these guidelines.[2]. Ignorance of guidelines may introduce undesirable effects in the software and may increase overall software effort [99]. Therefore, it is important to investigate into quality of bug reports and comments in Issue Tracking Systems.

Chromium is a popular open source web browser project based on which Google Chrome (also called Chrome) browser is built.[3] Chrome occupies 43% of the world wide browser market share as of March 2014.[4] Android is a popular mobile operating system based on the Linux kernel used for phones, tablets and other mobile devices.[5] The Android OS runs on 52% of the mobile devices as of February 2014.[6] The Issue Tracking Systems for both these projects are publicly available for download. Therefore, both these projects make an interesting test bed for our research due to their popularity, large data base and diversity. In this thesis, we look at quality indicators for bug reports and discussions on Google Chromium and Android Issue Tracking Systems. We also build systems and tools to help software maintenance professionals increase bug report quality during their daily maintenance tasks.

Twitter - Online Social Network

Twitter is one of the most used and immensely popular social network. Twitter is a micro-blogging website that allows registered users to share images, videos and text in short 140-character limit messages called *tweets*. Twitter reports that it has more than 200 Million monthly active users with more than 500 Million tweets posted everyday [24]. Previous research shows that Twitterers use Twitter to serve multiple purposes like - share their daily experiences, take part in conversations, share information (in the form of URLs, images, videos) and report & assimilate news [83]. Twitter provides many features to registered users to engage in multiple types of interactions with each other. Here, we briefly mention these features.

1. *Follow* – Twitter users connect with each other via a subscription based feature called the *Follow*. Twitter users can choose to 'follow' other fellow users to receive their tweets. For example, a user X can receive tweets from user Y if X 'follows' Y, indicating information flow from Y to X. The users who 'follow' user X are called *followers* of X and the users whom X follows are called *followees*.

2. *@-messages* – Twitter provides a feature to reply as well as send tweets to other users called *@-messages*. For example, a user X can reply or send a tweet to another user Y by writing @Y (where Y is a username) indicating an information flow from X to Y. Moreover, a user X can send an @-message to any other public user Z without *following* Z. An important distinction to note here is that the direction of information flow using *@-messages* is opposite to that of *Follow*.

[2] http://www.chromium.org/for-testers/bug-reporting-guidelines
[3] https://www.google.com/intl/en/chrome/browser/
[4] http://gs.statcounter.com/#browser-ww-monthly-201210-201304
[5] http://www.android.com
[6] http://www.dazeinfo.com/2014/04/16/apple-inc-aapl-claims-41-3-of-smartphone-market-but-android-enj

3. *Retweet or RTs* – Twitter allows its users to repost a tweet using the *ReTweet* (RT) button or by copying the previous message and prefixing the characters 'RT' to the tweet. An RT by a user signifies that the tweet is *interesting* to the user and hence, wants to re-share it. Similar to @-messages, a user need not follow another public user to RT his tweets.

4. *Lists* – Twitter users can categorize other users into buckets called *lists*. Users need not follow other users to create lists. Lists provide a great way to organize groups of users connected via a common theme. Lists can be publicly accessible to everyone or private to the owner.

Prior research shows severe gaps in the quality of content on Web 2.0 sites [60, 114, 115]. In this thesis, we explore the use of Twitter to fill in gaps over quality on social media. We also explore a new algorithm to extract homogeneous Twitter-based communities with the usage of significant interactions between users.

2.2 Content Quality on CQA websites

The two most central piece of units in any CQA website are Questions and Answers. Prior work has mostly focused on answer quality and best answer retrieval. In this section, We review literature about content quality on CQA websites. Finally, we discuss the reasons for our choice of CQA website for our experiments.

Answer Quality

Evaluation and prediction of answer quality has attracted wide spread attention in the IR research community. We review the literature on answer quality in CQA sites.

Machine learning approaches have been quite popular to predict answer quality. Burel *et al.* use thread-based features in conjunction to user-based and content-based features to build a machine learning classifier to find the best answers in CQA sites. They perform experiments on three diverse CQA sites – SAP Community Network (SCN)[7], ServerFault[8] and Cooking[9]. [46]. Their experiments show that discriminatory features vary across CQAs and are not generalizable. Toba *et al.* use a hybrid hierarchy-of-classifiers machine learning framework to model answer quality on Yahoo! Answers [125]. They argue that the quality of an answer directly depends on the type of the question. Therefore, a supervised classification method classifies each question in a QA pair into one six pre-defined categories – Definition, Factoid, Opinion, Procedure, Reason and YesNo. Next, each such category has another supervised classification algorithm for *answer quality* determination thus creating a hierarchy-of-classifiers. Their results show that good quality answers are long, well structured and contain appropriate references. Shah and Pomerantz propose a supervised

[7] `http://scn.sap.com/welcome`
[8] `http://serverfault.com/`
[9] `http://cooking.stackexchange.com/`

classification approach based on human assessed aspects and question-answer meta information to predict answer quality on Yahoo! Answers CQA [113]. Their study showed that human evaluators were able to assess factors affecting answer quality. But, these factors couldn't be encoded in the machine learning framework as they required to be put in context of the question. Hu *et al.* use textual and non-textual features in a multi-modal deep learning framework to estimate answer quality on Baidu Zhidao[10] and Yahoo! Answers [73]. Their experiments show that non-textual features are better estimators of answer quality than textual features. Further analysis of their approach shows that question related temporal features do not affect answer quality. Cai and Chakravarthy propose temporal features to predict answer quality in Yahoo! Answers and Stackoverflow [47]. Their experiment shows that the dynamic nature of CQA sites make temporal features more suitable to predict answer quality. Blooma *et al.* also study 17 answer quality indicators (social and content-based) in Yahoo! Answers [41]. Their experiments show that appraisal features based on content like positive votes, completeness, presentation, reliability and accuracy are the best indicators of answer quality. Ginsca and Popescu investigate the relevance of user profile and activity features for answer quality on Stackoverflow [62]. They show that user profiles features like location, name, avatar and community involvement like profile views, votes are highly correlated with answer quality.

Apart from machine learning based approaches, some solutions have taken an Information Retrieval view of the problem. Jeon *et al.* propose Maximum Entropy and Kernel Density Estimation(KDE) approach in conjunction with non-textual features to predict answer quality on Naver, a Korean CQA website [76]. They incorporate quality scores into a language-modeling based retrieval model and perform experiments to show significant improvement in question-answer retrieval performance. Zhou *et al.* propose an answer quality assessment method to benefit question routing on Yahoo!Answers [137]. They develop a probabilistic framework to predict answer quality score based on prior question interest of the scores and answer structures. Sakai *et al.* propose evaluation methods based on graded-relevance IR metrics to find the best answers on Yahoo! Chiebukuro (Japanese Yahoo! Answers) [110].

Finally, we also observe some empirical studies to understand answer quality. Shachaf performs a comparison of answer quality on four websites – Askville[11], WikiAnswers[12], Wikipedia Reference Desk[13], and Yahoo! Answers [56]. The study reports that there are appreciable answer quality variations across CQA sites and popularity of a CQA site has no direct correlation on answer quality. Harper *et al.* study answer quality indicators on three CQA sites - Google Answers, AllExperts and Yahoo! Answers [68]. Their study shows that paid CQA sites perform better than free CQAs but these free CQAs have considerable variance in answer quality. Overall, all CQA sites were primarily driven by a set of motivated community contributors. Yao *et al.* perform an empirical study on the quality of answers on Stackoverflow [134]. Their empirical results show that answer quality on Stackoverflow has a positive correlation with question quality. Yao *et al.*

[10] zhidao.baidu.com

[11] http://askville.amazon.com/Index.do

[12] http://wiki.answers.com/

[13] http://en.wikipedia.org/wiki/Wikipedia:RD

perform an empirical study on the quality of answers on Stackoverflow [134]. Their empirical results show that answer quality on Stackoverflow has a positive correlation with question quality.

Research Gap – Question Quality?

However, all the previously discussed approaches focus on answer quality on large scale CQA websites. Question quality is paramount because prior work shows that answer quality directly depends on question quality [134]. Low quality questions have a direct impact on user experience, question retrieval, question recommendation and hence, it is important to maintain high question quality [93]. Quality control of content on such large scale community driven collaborative systems is a research challenge [27]. Questions and answers form an integral part of any CQA website and therefore, it is important to have quality checks in place for both questions and answers.[14] Li *et al.* analyze factors affecting question quality and propose a Mutual Reinforcement-based Label Propagation approach to predict question quality in Yahoo! Answers [87]. To the best of our knowledge, this is the only study on question quality in CQA sites. Our literature review suggests a gap in understanding about quality of questions on CQA websites. In this thesis, we attempt to address this gap and provide perspective on quality of questions on Stackoverflow.

Why study StackOverflow?

Stack Overflow is a popular collaborative Q&A website used by programmers all over the world to seek answers to programming related questions [93]. Stack Overflow is a free and open website and has 2M+ registered users with 7.1M questions and 12M answers. The underlying theme of Stack Overflow is programming-related topics and the target audience are software developers, maintenance professionals and programmers. Besides being a question-answer website, Stack Overflow has evolved into a knowledge base for programming related tasks [30]. Therefore, Stack Overflow has attracted increasing attention from different research communities like software engineering, human computer interaction, social computing and data mining [32, 38, 43, 101, 105]. Researchers have mined the Stack Overflow knowledge-base to extract unique insights for various core and ancillary programming tasks like building crowd sourced API documentation, API usage obstacles, innovation diffusion via URL link sharing, mobile development issues, improvement of bug tracking systems and programming topic trends [38, 48, 65, 88, 92, 103, 131].

Nasehi *et al.* analyze questions on Stack Overflow to understand the quality of a code example [96]. They find nine attributes of good questions like concise code, links to extra resources and inline documentation. Wang and Godfrey analyze iOS and Android developer questions on Stack Overflow to detect API usage obstacles [131]. They used topic models to find a set API classes on iOS and Android documentation which were difficult for developers to understand. Asaduzzaman *et al.* analyze unanswered questions on Stack Overflow and use a machine learning classifier to predict such questions [33]. They observe certain characteristics of unanswered questions

[14] `http://meta.stackoverflow.com/questions/252506/question-quality-is-dropping-on-stack-overflow?cb=1`

which include vagueness, homework questions etc. Allamanis and Sutton perform a topic modeling analysis on Stack Overflow questions to combine topics, types and code [28]. They find that programming languages are a mixture of concepts and questions on Stack Overflow are concerned with the code example rather than the application domain. Stackoverflow has made its entire data publicly available for research purposes. Considering the structure, content and availability of data Stackoverflow is a very good potential testbed to perform our experiments.

2.3 Content Quality Systems

Quality maintenance is an important process in software maintenance. During the process of software maintenance, professionals browse the web to search for solutions or create specific road maps [40, 140]. The web has become a valuable assistant for software maintenance. However, currently maintenance engineers need to switch *context* between the Internet and software maintenance tools like Issue Tracking Systems for their daily tasks. In this thesis, we look at ways to bridge such a switch of *context* by building systems to allow integration of Internet tasks with the maintenance toolbox. We posit that integrating such systems will drive maintenance professionals to contribute better content thus, enhancing content quality. We build on our research from the previous part of the thesis to identify two daily work flows in order to develop these systems. Concretely, we look at two critical resources used by maintenance professionals – (1) Code Development Environment and (2) Web browsing references aka URLs. In this section, we review literature for quality enablers during the software maintenance process.

Systems for Code Development Environment and Web Search Integration

Bacchelli *et al.* describe an Eclipse plug-in called as Seahawk which integrates Stackoverflow with Eclipse Integrated Development Environment (IDE) [35]. Their work is motivated by the need of seamlessly accessing Stack Overflow data within the IDE so that there is no context switching between the IDE and Stack Overflow website. Seahawk Eclipse plug-in has a feature that allows developers to search and retrieve Stack Overflow Q&A within the IDE and also link Stack Overflow discussion threads to the source code loaded within Eclipse [35]. Cordeiro *et al.* propose a system which integrates Eclipse IDE (Integrated Development Environment) and Stackoverflow.com (Q&A web resource) [51]. They describe a prototype (implemented as an Eclipse Plug-In) which monitors occurrences of exception stack traces in Eclipse console view and automatically searches a Q&A website (by providing keywords from the exception stack traces) to retrieve results ranked according to the information need (derived from the stack trace). Cordeiro *et al.* experiments with a context-based recommendation approach and demonstrates that the proposed solution outperforms a simple keyword-based method [51].

Brandt *et al.* believe that integration of code editors and search tools has significant value and describe a tool called as Blueprint which is implemented (integrating web search) as an extension to a development environment [44]. Their system Blueprint (accessing online example code from within the development environment) is integrated with Adobe Flex Builder, uses Adobe

Community Help search API and indexes Flex-specific content. They present experimental results which demonstrates that a integration of development environment with web-search is helpful for the programmers (users were able to search relevant example code considerable faster in contrast to searching information through a standard browser as a separate standalone application) [44]. Goldman present a system called as Codetrail that connects the Eclipse IDE and Firefox Web Browser motivated by the need to integrate web resources into the programming workflow and into the project itself [63]. Codetrail automates some of the manual interactions of developers with web resources and consists of features such as automatic detection and connection of documentation, creation of links, or bookmarks, from code in the development environment to relevant web sites [63]. Hartmann *et al.* describe a IDE extension called as HyperSource that tracks developers activities in the IDE (such as source code edits) and associates web-pages with the code-edits [69]. HyperSource associates browsing histories with source code edits and displays sets of Web pages that were read while code was edited [69]. Sawadsky *et al.* describe a tool called as Fishtail which is implemented as a Eclipse IDE plug-in to support programmers in discovering code examples and documentation on the web relevant to their current software engineering task [112].

Systems for Discussions in Issue Tracker and Software Forums

Gottipati *et al.* present a method to infer semantic tags (answers, clarifying answers, clarifying questions, positive and negative feedback) of comment posts in software forum threads and exploit the inferred tags to retrieve relevant answer posts from a give query [67]. Dit et al. present a method to automatically make connections between new comments and previous comments. The system also recommends comments to a user based on their comments (improving readability) [55]. Lotufo et al. propose a system (called as BugBot) that promotes information instead of conversation and shows the effectiveness of the approach using empirical analysis [90]. Solution approaches and techniques for bug report summarization have been proposed in the past [106, 91]. Lotufo et al. present a bug report summarization approach (estimates the attention a user would hypothetically give to different sentences in a bug report, when pressed with time) to facilitate bug report digestion [91]. Software maintenance process also involves Internet search to look at information to help daily tasks.

The study by Correa *et al.* reveals that discussion forums on issue tracking systems contain links to several Web 2.0 platforms, web applications and social media websites. They present a recommendation algorithm to retrieve relevant Q&A artifact from StackOverflow motivated by the need to facilitate developers find relevant discussion (on a Q&A website for programmers) to a given issue report [54]. Brandt *et al.* propose a system called as Blueprint which integrates code editors with search tools [44]. Gomez *et al.* conduct a study of innovation diffusion through link sharing on StackOverflow (a popular community driven Q&A website) [64]. Their study reveals that link sharing (such as official documentation, blog post, Q&A post, Wiki, forum post and code repository) is a significant phenomenon on StackOverflow and discover that developers disseminate software development innovations by sharing URLs with other developers [64].

Research Gaps

Based on our literature survey, we find two research gaps with respect to the objectives of our thesis.

Integration of Issue Tracking Systems with CQA

Zagalsky *et al.* describe a code search and a recommendation tool called as Example Overflow which mines information present in Stack Overflow (Q&A website for programmers) [135]. Their work is motivated by the need to minimize context switch between development environment and code-search tools. They perform a case-study on jQuery related Q&A and prove the effectiveness (relevance of search results and number of context switches required) of code example search from Stack Overflow [135]. Our survey shows that StackOverflow is also used by developers during the software maintenance process [54]. Therefore, we aim to build a system to integrate CQA sites with Issue Tracking Systems to act as an quality enabler during the software maintenance process.

Web or URL Reference Management in Issue Tracking Systems

The study by Fourney *et al.* reveals that askers and answerers of technical forum questions typically conduct extensive online research before composing their posts and develop a tool called as CiteHistory (a web-browser plugin) that simplifies the process of including relevant search queries and URLs as bibliographic supplements to forum posts [57]. Our analysis reveals that developers browse the web and share links in Issue Tracking Systems [52]. Therefore, we build a browser plugin – *Samekana* – to ease the process of managing references during bug fixing and feature enhancement tasks.

2.4 Information Retrieval Enhancement

There have been various studies on quality maintenance in social media. We look at prior literature on social media with respect to quality enabling search and mining user generated content. Concretely, we review literature on two core aspects of social media which helps effective information retrieval – (i) social network interactions and (ii) tag annotations on shared media.

Community Discovery on Social Media

Over the past decade, community detection has attracted research attention all over the world. Various algorithms have been proposed to address issues in this area. Santo and Fortunato have performed an in-depth survey in the problem of community detection in graphs [111]. Tscherteu and Langreiter propose a heuristic based "bottom-up" snowball algorithm to detect topic-centric communities on Twitter [128]. Recently there have been many approaches to combine content and link structure for community detection on social networks [104, 109, 133]. Pathak *et al.* demonstrate a *Community-Author-Topic* (CART) Bayesian generative model which combines link

and content for community extraction [104]. Yang *et al.* demonstrate a split approach which uses a *conditional* model for links in a network and *discriminative* model for the content [133]. Sachan *et al.* propose Topic User Recipient Community Model (TURCM) which uses a generative process based on the number of messages sent between users in the social network and the content being discussed [108, 109].

Tag Recommendation on Social Media

Previous studies describe techniques for automatic tag annotation[60, 114, 115, 116], classification[100], clustering[39] and refinement [89][139] in various social media services like bibsonomy, online bookmarks, images, songs and videos. Overell *et al.* present a generic method for classifying tags into semantic categories using third party open content resources, such as Wikipedia and the Open Directory apply their system for classifying Flickr tags [100]. Zhao *et al.* present a method to annotate web videos by performing experiments on cross sources (annotating Google and Yahoo! videos using YouTube videos) [136]. Stewart *et al.* present a method for cross-tagging where tags from one social system are recommended in order to automatically annotate resources in another social system. They perform experiments on Blogger (blogging site) and Last.fm (a social music site) [117].

Research Gaps

In context to community discovery and tag recommendation, we identify research gaps in prior literature.

iTop – Interaction Based, Local Community Discovery

Our approach takes a new direction to community detection on social networks with respect to these previous approaches. Previous algorithms require the complete graph to be loaded into memory for community detection [104, 109, 133]. *iTop* snowballs from a true-positive seed on a topic and discovers concentric communities based on a structural objective function (local modularity). *iTop* targets application scenarios which need an effective way to find communities around a specified topic. For example, a digital marketing team for a company *X* would like to know the communities around *X* and not around other unrelated topics. Moreover, with the advent of *Big Data*, there has been a rise in, *On Demand* data services like *Data Sift* [15] which have a *pay-as-use* policy to access their data. Previous approaches to detect topic-centric communities around Twitter do not take these factors into account while developing their solutions. In contrast, *iTop* takes an input topic and discovers concentric communities around these nodes and avoids visitation to nodes and content in the social network which are not of interest for our input topic. Such an approach would reduce computational and data costs and greatly benefit security analysts working at law enforcement agencies and digital marketing analysts to find pertinent communities of their interest. In addition, we take into account heterogenous relations between users like

[15] http://datasift.com/

Retweet, Replies and Mentions while discovering these communities. Previous approaches use an explicit link (*follower*) in the social network or limited user interaction to define the social graph of the network [108, 109]. However, we incorporate multiple interactions like retweets, replies and mentions and form a directed weighted graph by the frequency of their interaction to perform our experiments.

Mining Tweets for Tag Recommendation

Tags play an important role in search and retrieval in multimedia systems. However, tags are user supplied and these tags may be incomplete, missing or irrelevant. Therefore, it is important to develop automatic algorithms to ensure tag quality. In this work, we develop a complementary technique to recommend tags on social media. Concretely, the proposed method consists of mining *tweets* to semantically enrich web-resources on image, video and audio sharing websites. We perform experiments on data crawled from Twitter and Flickr, Photobucket, Dailymotion, YouTube, SoundCloud. Experimental results demonstrate that a *Wisdom of Crowd* mashup framework can be an effective solution for tag recommendation.

2.5 Conclusion

In this chapter, we reviewed literature according to the objectives of our thesis in three areas – (1) Low Quality Content on CQA sites, (2) Content Quality Systems for Software Maintenance Tasks and (3) Information Retrieval Enhancement on Social Media. We also discussed research gaps in each of the three dimensions in context of the contributions of this thesis. With this background, we now move on to the next chapters which detail on concrete solutions to address these research gaps with experiments and results. The first two chapters address low quality content on CQA sites and the next two chapters look into building systems and techniques to enhance quality during the software maintenance process. The subsequent two chapters look into enhancing information retrieval algorithms on social media. Figure 2.1 shows the overview of our three pronged approach to look into the problem of content quality on Web 2.0 services.

Figure 2.1: shows the overview of our three pronged approach to look into the problem of content quality on Web 2.0 services

Chapter 3

Closed Questions on StackOverflow

Analysis and detection of low quality content on Web 2.0 services is a research challenge. In this chapter, we make the first attempt to address the problem of low quality questions on CQA websites. Concretely, in this chapter we look at one type of low quality questions on Stackoverflow – *closed* questions. We perform an in-depth study of *closed* questions on Stackoverflow and build a system to detect a *closed* question at creation time.

3.1 Introduction

Research Motivation and Aim

Stack Overflow guidelines clearly outline categories of questions which are deemed unfit for its Q&A format. Opinion-based questions and questions which have a tendency to generate discussions rather than answers are categorically considered inappropriate. Some examples of such questions include (but not limited to) homework questions, product or service recommendations, non-programming related and polls. Questions on Stack Overflow which do not fall into one of the pre-defined set of guidelines are marked 'closed' via a community-based voting system. A question can be marked as 'closed' for five reasons – *duplicate, off-topic, subjective, not a real question* and *too localized*. Section 3.2 contains a detailed discussion on the procedure to mark a question 'closed' and its sub-categories. Figure 3.1 shows an example of a 'closed' question on Stack Overflow on account of being *Too Localized*.

A question is primarily marked 'closed' either due to low quality or due to irrelevance to the Stack Overflow CQA platform. The decision to 'close' a question lies completely on the shoulders of experienced users and community moderators via a systematic voting process. Due to exponential growth of Stack Overflow user base, there has been a steady increase in the workload on moderators. The process of marking a 'closed' question also requires multiple context switches [25]. Despite the existence of vibrant experienced users and self-motivated community moderators, Stack Overflow faces a continuous ongoing challenge to maintain quality of questions on their website. Therefore, it is important to analyze and study the phenomena of 'closed'

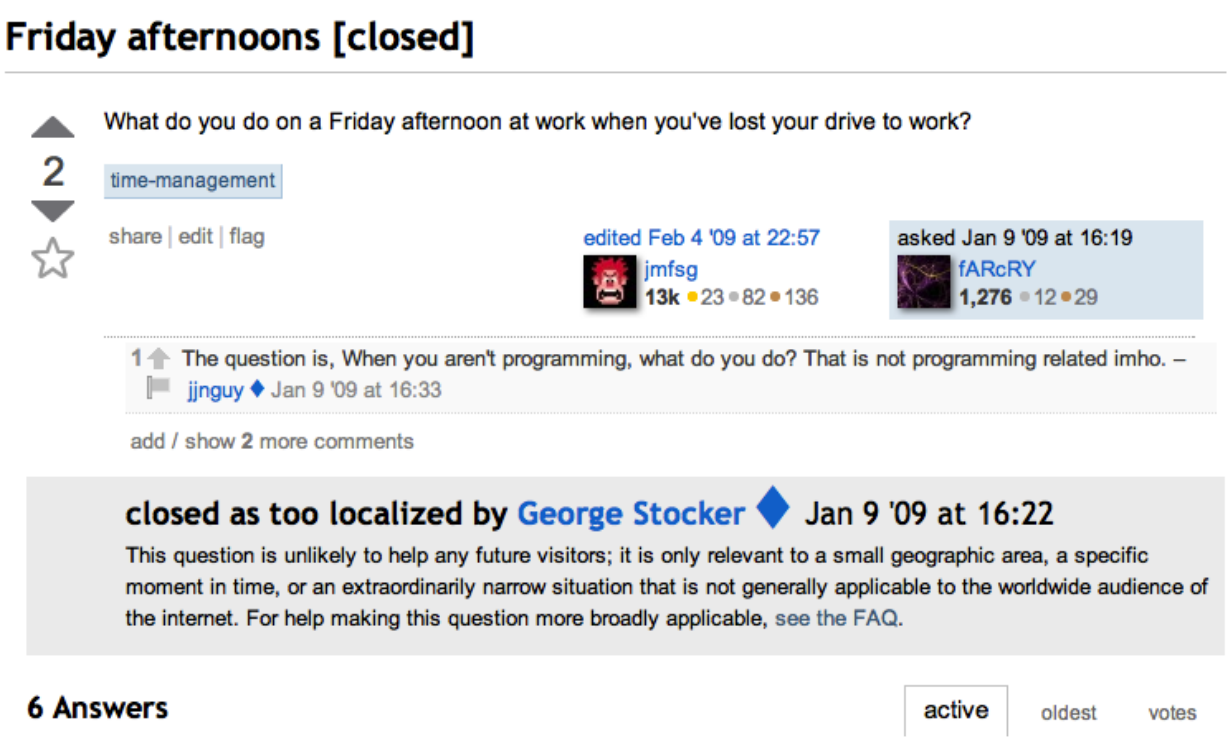


Figure 3.1: shows a screenshot of question marked 'closed' on Stack Overflow on account of being *Too Localized*.

questions in order to gain historical insights which can help make the future plan-of-action.

The goal of Stack Overflow is to have a knowledge base of question-answers on programming related topics. A 'closed' question is a direct feedback to the question asker that her question may be unfit or needs improvement in its current form. A system to predict a 'closed' question at post creation time can serve as an early feedback mechanism on question quality to the question asker. Such a system would also help community moderators to identify and mark 'closed' questions. Therefore, prediction of a 'closed' question at post creation time has two distinct benefits

1. Early feedback mechanism to the question asker which would serve as an indicator that her question may be deemed unfit for Stack Overflow Q&A format
2. A complementary mechanism to aid community moderators in their daily moderation tasks to 'close' unfit questions

In this work, we present the first study of 'closed' questions on Stack Overflow. We download 4 years of publicly available data which contains 3.4 Million questions. We first analyze and characterize the complete set of 0.1 Million 'closed' questions. Next, we use a machine learning framework and build a predictive model to identify a 'closed' question at the time of question creation. One of our key findings is that despite being marked as 'closed', *subjective* questions contain high information value and are very popular with the users. We observe an increasing

trend in the percentage of closed questions over time and find that this increase is positively correlated to the number of newly registered users. In addition, we also see a decrease in community participation to mark a ‘closed’ question which has led to an increase in moderation job time. We also find that questions closed with the *Duplicate* and *Off Topic* labels are relatively more prone to reputation gaming. Our analysis suggests broader implications for content quality maintenance on CQA websites. For the ‘closed’ question prediction task, we make use of multiple genres of feature sets based on - user profile , community process, textual style and question content. We use a state-of-art machine learning classifier based on an ensemble framework and achieve an overall accuracy of 70.3%. Analysis of the feature space reveals that ‘closed’ questions are relatively less informative and descriptive than non-‘closed’ questions. To the best of our knowledge, this is the first experimental study to analyze and predict ‘closed’ questions on Stack Overflow.

3.2 ‘Closed’ Questions on Stack Overflow

In this section, we discuss details on *who*, *how* and *why* questions are closed on Stack Overflow. We also briefly outline *what happens* once a question is ‘closed’ and mention the community process rules to mark a question as ‘closed’. Figure 3.2 summarizes the details of important aspects of ‘closed’ questions on Stack Overflow.

What is a ‘closed’ a question?

A question can be ‘closed’ on Stack Overflow if it is deemed unfit for its Q&A format [12]. A ‘closed’ question can not be answered but edits on previously posted question-answers and comments are permitted (subject to appropriate edit privileges). Question-answers can also be voted upon and are counted towards reputation points of users as well as badges.

Who can ‘close’ a question?

Experienced users and community moderators can cast a vote to ‘close’ a question. Stack Overflow users with 3,000+ reputation points and community moderators (also called ◆ moderators) can vote for the same. In addition, users with at least 250 reputation points can vote to ‘close’ their own question. The *Who* block of Figure 3.2 corresponds to the aforementioned details.

How are questions ‘closed’ ?

A question is automatically marked ‘closed’ if it receives 5 ‘close’ votes. However, ◆ moderator ‘close’ votes are final and binding i.e. if a ◆ moderator decides to cast a ‘close’ vote the question is ‘closed’ immediately [21]. One can only vote once to ‘close’ a question. The *How* block of Figure 3.2 corresponds to this process.

Why are questions 'closed'?

According to Stack Overflow guidelines, a question is 'closed' on Stack Overflow if it falls into one of the following five categories [12]:

1. **Exact Duplicate** – contains similar content to previously posted questions
2. **Off Topic** – unrelated to programming scope as defined by Stack Overflow
3. **Subjective (Not Constructive)** – more likely to generate debates, discussions instead of answers
4. **Not a Real Question** – ambiguous, vague questions which do not have answers
5. **Too Localized** – relevant to a very small geographic location, software or community

The *Why* block of Figure 3.2 corresponds to this section.

What happens to a 'closed' question?

A 'closed' question can be 'reopened' if the question is improved from its current form. The 'reopen' voting procedure is similar to the 'close' procedure. However, if the questions are very poor in quality and beyond improvement, then they are *deleted* from Stack Overflow [12]. The *What* block of Figure 3.2 corresponds to this section.

3.3 Characterization Study of 'Closed' Questions

In the first part of our work, we perform a characterization study of 'closed' questions on Stack Overflow.

Dataset Description

Stack Overflow provides all user-generated content on its website for download under the *Creative Commons Attribute-ShareAlike* license [34]. We download Stack Overflow website data from the Stack Exchange August 2012 data dump provided by Stack Overflow which contains all data between July 31st, 2008 (the genesis of Stack Overflow) to August 31st, 2012 [22]. Table 3.1 outlines basic statistics for Stack Overflow August 2012 dataset used in our characterization study. The statistics show that Stack Overflow is a very popular programming CQA with 1.29M registered users, 3.4M questions and 6.8M answers.

In this work, we concentrate on 'closed' questions on Stack Overflow i.e. questions which are deemed unfit and therefore low quality given the context. We extract all questions from the dataset which have been marked 'closed' at least once. We find that approx. **3% (0.1 Million)** questions are marked 'closed' on Stack Overflow between August 2008 to August 2012. We use this data of

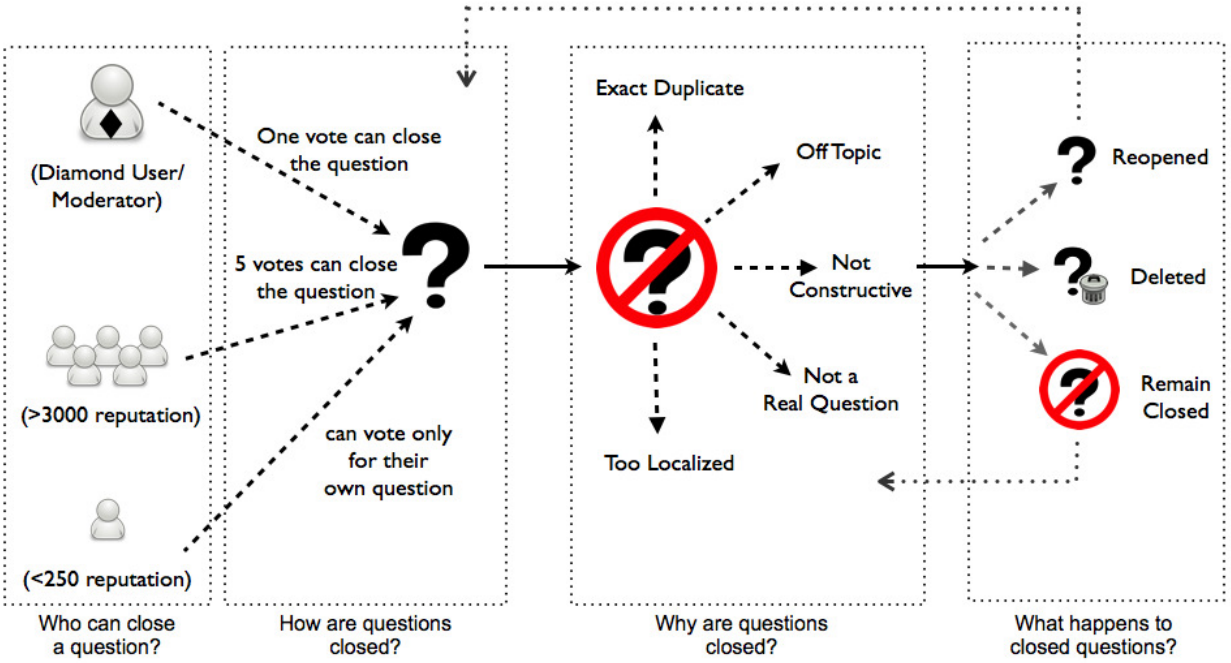


Figure 3.2: depicts *who*, *how* and *why* questions are marked 'closed' on Stack Overflow.

Table 3.1: Stack Overflow August 2012 dataset statistics

Users	1.29M (625k askers, 443k answerers)
Questions	3.4M (62.21% with accepted answers)
Answers	6.8M (31.33% marked as accepted)
Votes	27.5M (72.35% positive, 6.81% favorites)
Ratio of Answers to Questions	2.16

102,993 'closed' questions to conduct our characterization study and report our findings. Table 3.2 contains details on 'closed' questions in Stack Overflow. [1]

Based on the data, we can make two observations – (1) Stack Overflow maintains a very good signal-to-noise ratio as reported in previous work [93] and (2) Despite the presence of vibrant community and structured guidelines, users do post questions which are unfit for the website. A question can be closed on Stack Overflow for five reasons - *duplicate, off-topic, subjective, not a real question* and *too localized*. Figure 3.3 shows a pie-chart which depicts the distribution of 'closed' questions on different sub-categories or reasons. *Not a Real Question* and *Duplicate* categories are the most common reasons to close a question while *Too Localized* is the least common reason.

[1]Prior to June 2011, 'Close Votes' expired 4 days after their cast and are deleted from the dataset published by Stack Overflow. This information is available only if a question is closed successfully.

Table 3.2: Statistics of 'Closed Questions' in Stack Overflow from August 2008 to August 2012.

	2008	2009	2010	2011	2012	Total
Closed Questions	3.8%	1.52%	1.77%	3.33%	3.82%	102, 993 (2.98%)
Closed Votes	0.03%[3]	0.25%[3]	0.75%[3]	2.21%	3.9%	570,418[3] (0.2%)
Ratio of Answers to Questions	8.0	5.93	3.11	1.92	1.55	1.92

Temporal Distribution Analysis

We analyze the presence of 'closed' questions on Stack Overflow over a 48-month time window between August 2008 to August 2012. Figure 3.4 depicts the ratio of 'closed' questions to total questions over this time period. Overall, we find an increasing trend of the percentage of 'closed' questions in each category i.e. we find that the number of questions 'closed' over time has an upward curve. We also see that the most common categories of 'closed' questions over 48-months are *Exact Duplicate* and *Not a Real Question*. Both these categories dominate in presence over the others across time. We perform qualitative analysis of some sample questions in our dataset to understand this pattern. The high ratio of the *Exact Duplicate* category may be due to the problem of question retrieval on Stack Overflow i.e. users are unable to efficiently locate questions which are similar to the actual problem they are faced with. The presence of such a high ratio may also be due to lethargic users who do not perform adequate searches before posting a question. Similarly, the high percentage of *Not a Real Question* category may be due to newly registered users who are yet to understand the scope, structure and guidelines of Stack Overflow. Overall, we see a sharp increase in the ratio of 'closed' questions after January 2011.

Effect of New Registered Users

Questions are marked 'closed' on Stack Overflow if they are considered unfit for its Q&A format. Intuitively, newly registered users on the website may be indolent to existing guidelines and may ignore them in their anxiety to get a solution to a problem. Therefore, we try to understand the impact of newly registered users on the presence of 'closed' questions on Stack Overflow over time. Figure 3.5 shows the distribution of – (1) number of newly registered users and (2) percentage of 'closed' questions on Stack Overflow – over a 48-month period between August 2008 to August 2012. In addition, it also depicts the corresponding *Pearson Correlation Coefficient* (PCC) between the two distributions (cumulative) at each time interval. PCC calculates the linear dependence between two distributions and outputs a value between +1 (positive correlation) to -1 (negative correlation). Figure 3.5 shows a high correlation between the number of newly registered users and percentage of closed questions. We stress that the calculated correlation coefficient is between new registered users and the **percentage of closed questions** (and not the total number of closed

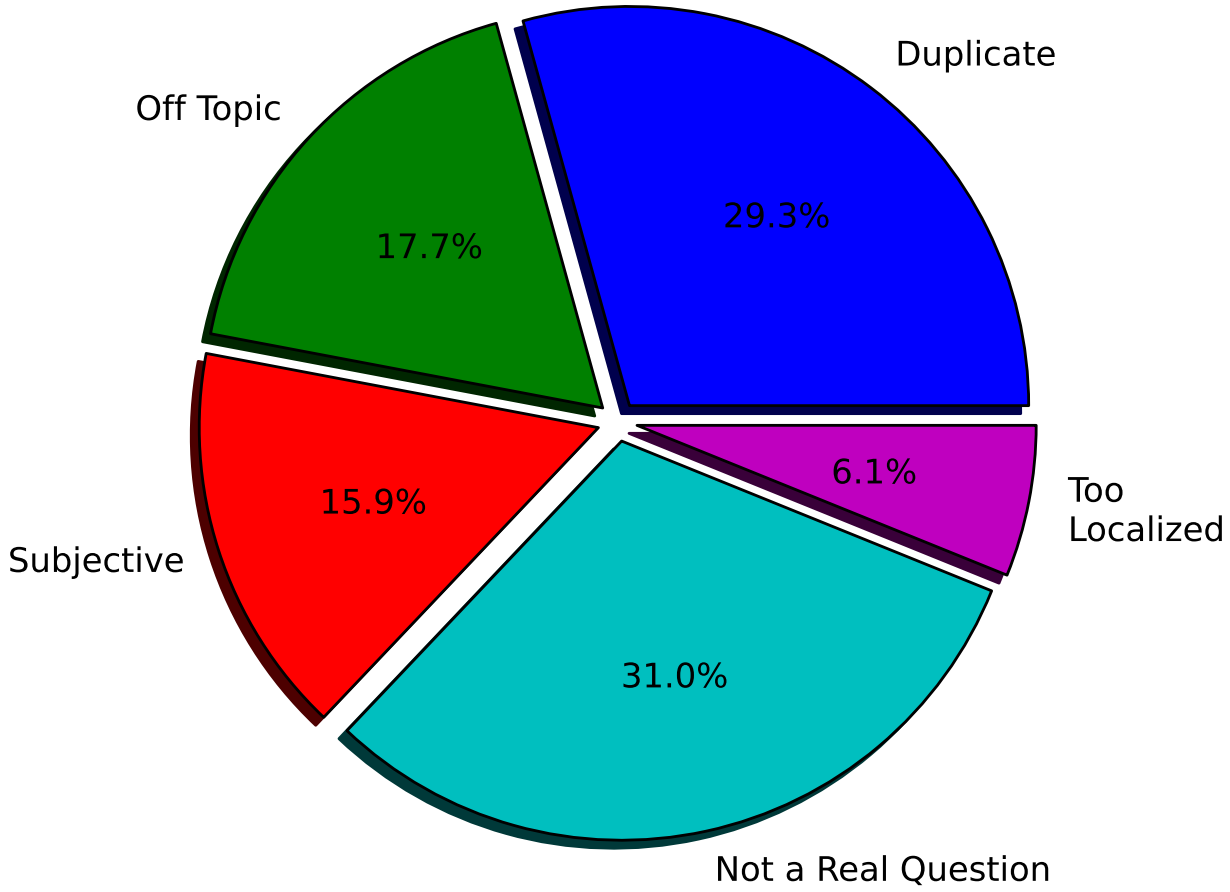


Figure 3.3: shows the distribution of all five sub-categories of closed questions in our dataset.

questions) over time. The PCC value is +0.95 which indicates a very high correlation between the distributions with an extremely high confidence interval (p-value < 0.01). The PCC shows that newly registered users may have an immediate impact on low quality content. Here too, we find a sharp rise in PCC after January 2011.

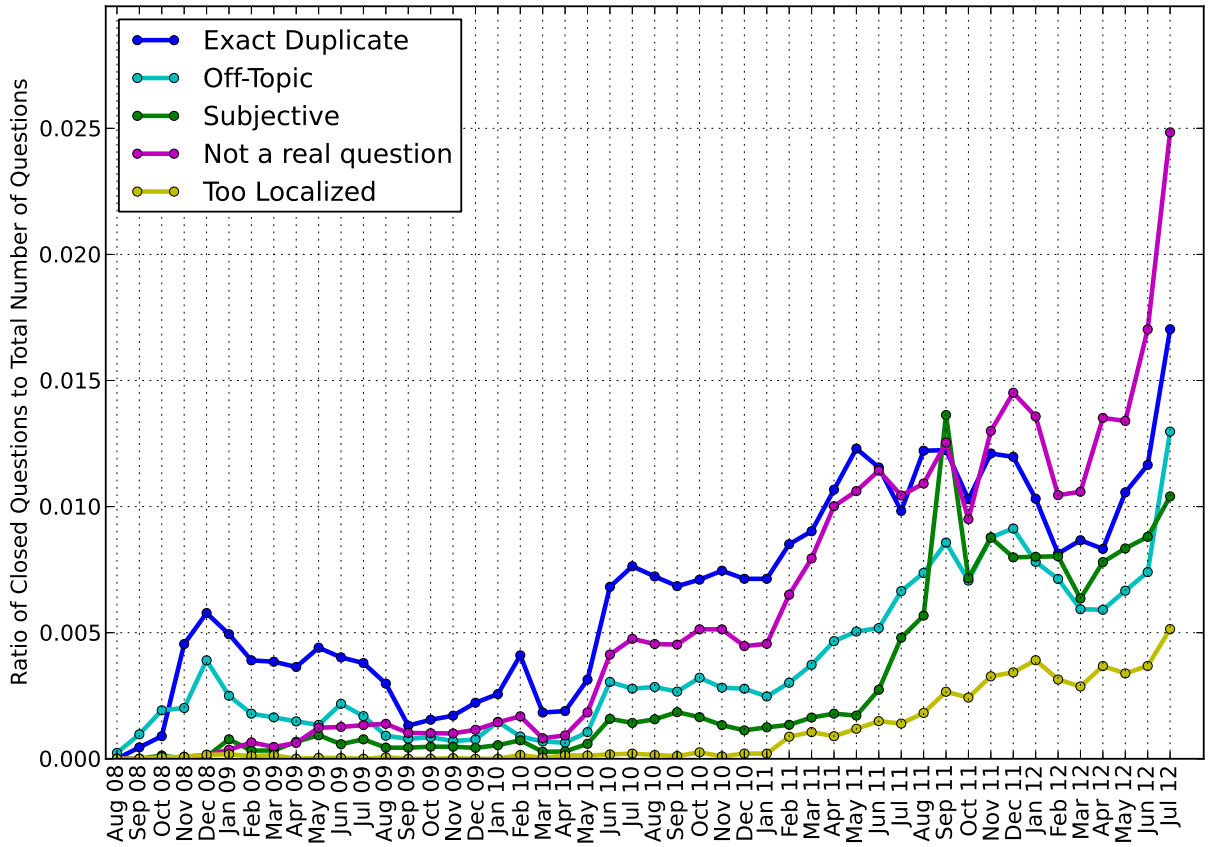


Figure 3.4: shows the temporal distribution plot of the ratio of 'closed questions' to total questions over a 48-month period between August 2008 to August 2012 for each sub-category.

Community Participation

Stack Overflow follows a well defined community based voting procedure to evaluate a question before closure. We analyze these voting patterns to understand *community participation* of experienced users and community moderators to weed out low quality content on the website. We recall that users with 3,000+ reputation points and ◆ moderators can cast a vote to close a question. A question is automatically 'closed' if it reaches 5 votes but a vote from a ◆ moderator is binding and hence, immediately closes a question. Therefore, a question can be closed with any number of 'close' votes between 1 to 5. Figure 3.6 shows the temporal distribution of 'close' votes on Stack Overflow between August 2008 to August 2012. Table 3.3 shows the distribution of number of 'close votes' on closed questions. A significant percentage(≈27%) of questions are closed due to a single ◆ moderator vote. More than 40% of questions require ◆ moderator intervention to close a question. We also observe a rise in the percentage of questions being closed only by ◆ moderators over time. Simultaneously, we see a decrease in percentage of questions being closed by experienced users viz. users with 3,000+ reputation points. This shows that community participation to

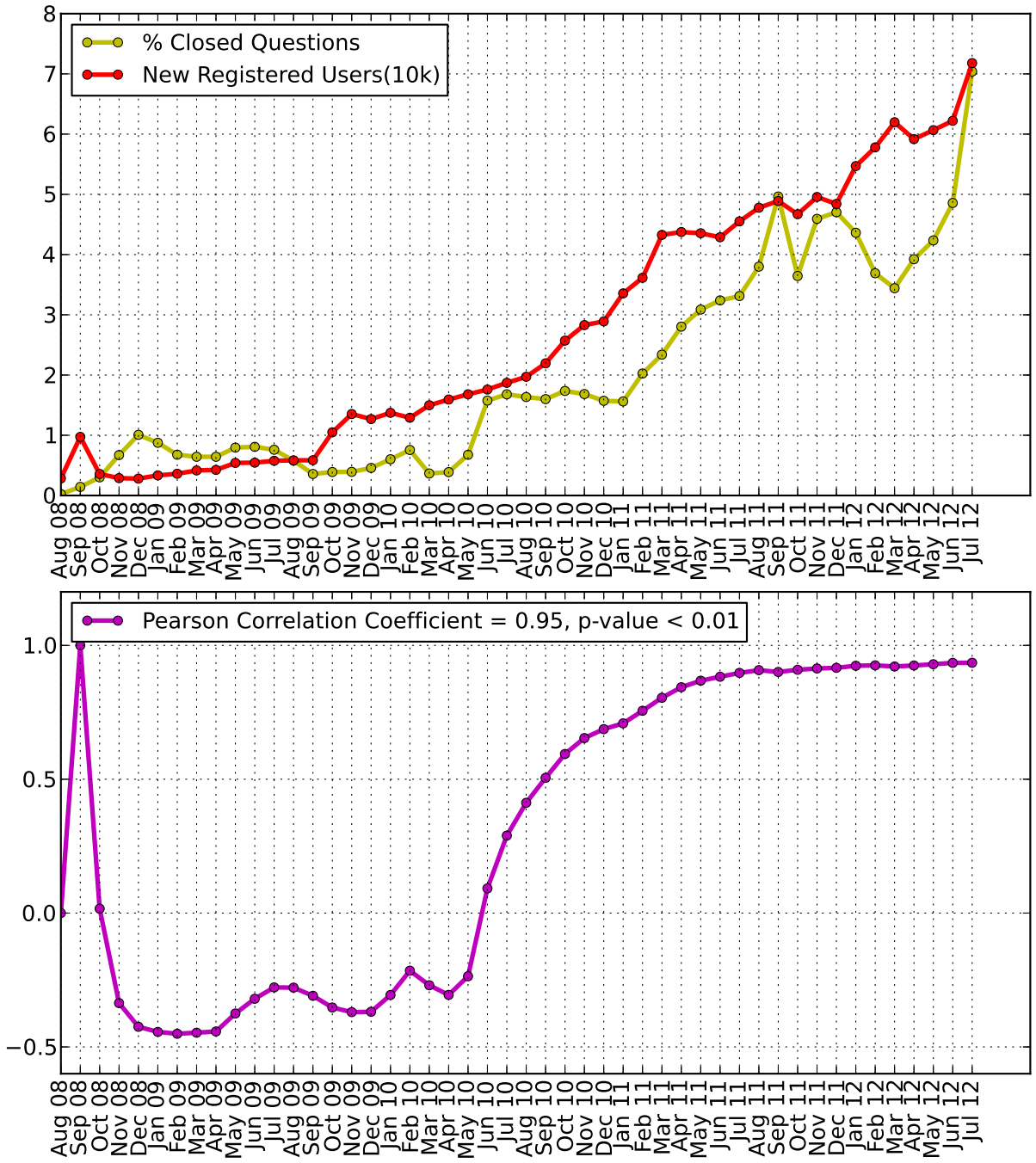


Figure 3.5: shows the temporal distribution plot of the percentage of 'closed questions' and newly registered users over a 48-month period from August 2008 to August 2012. In addition, the figure also shows correlation between both distributions.

close questions is on a decline which has led to an increase in work load for ♦ moderators on this front. A ♦ moderator on Stack Overflow has also confirmed an increase in moderation work load over the years [25]. Stack Overflow has only 16 ♦ moderators for their website out of which 13 have been elected and 3 have been appointed [23].

Table 3.3: shows the 'Close Vote' Distribution on 'Closed' questions posted between August 2008 and August 2012. 45% questions have at least one ♦ moderator vote and 26.5% of questions are closed by a single ♦ moderator vote.

Votes	Closed Questions
1-vote	27,390 (26.59%)
2-votes	9,037 (8.77 %)
3-votes	5,436 (5.28%)
4-votes	4,030 (3.91%)
5-votes	57,117 (55.44 %)
Total	102,993

We now analyze the 'close vote' patterns across each category of closed questions. Figure 3.7 shows the 'close vote' distribution for each sub-category of closed questions on Stack Overflow between August 2008 to August 2012. We see a strong community participation on *Duplicate*, *Off Topic* and *Not a Real Question* categories. On the other hand, *Subjective* and *Too Localized* categories require a high amount of ♦ moderator intervention. We argue that the community participation behavior may be so because *Duplicate*, *Off Topic* and *Not a Real Question* questions are low hanging fruits and easy to detect. The *Subjective* category sees an equal community and ♦ moderator participation. The *Too Localized* category sees a higher ♦ moderator intervention. Since, the presence of this category is very low in our dataset, such behavior may be primarily due to low traction owing to the difficulty of identification of such questions during normal daily usage of the website.

Content Analysis

We now characterize the content of 'closed' questions on Stack Overflow based on question tittle, question content, code snippets and topics.

Question Title, Body and Code Snippet

Since, 'closed' questions are unfit for Stack Overflow – the presence or absence of code snippets may reveal insights about 'closed' questions. Overall, $\approx 31\%$ of 'closed' questions contain code snippets and hence, questions are 'closed' even if they contain source code. We analyze the presence of code snippets across each category to check if there are relative differences across categories. Figure 3.8 (left-top) shows the percentage of questions which contain code snippets for each category. We find that *Too Localized* and *Exact Duplicate* category contains a large number of questions which have code snippets in them. The *Exact Duplicate* category by definition contains

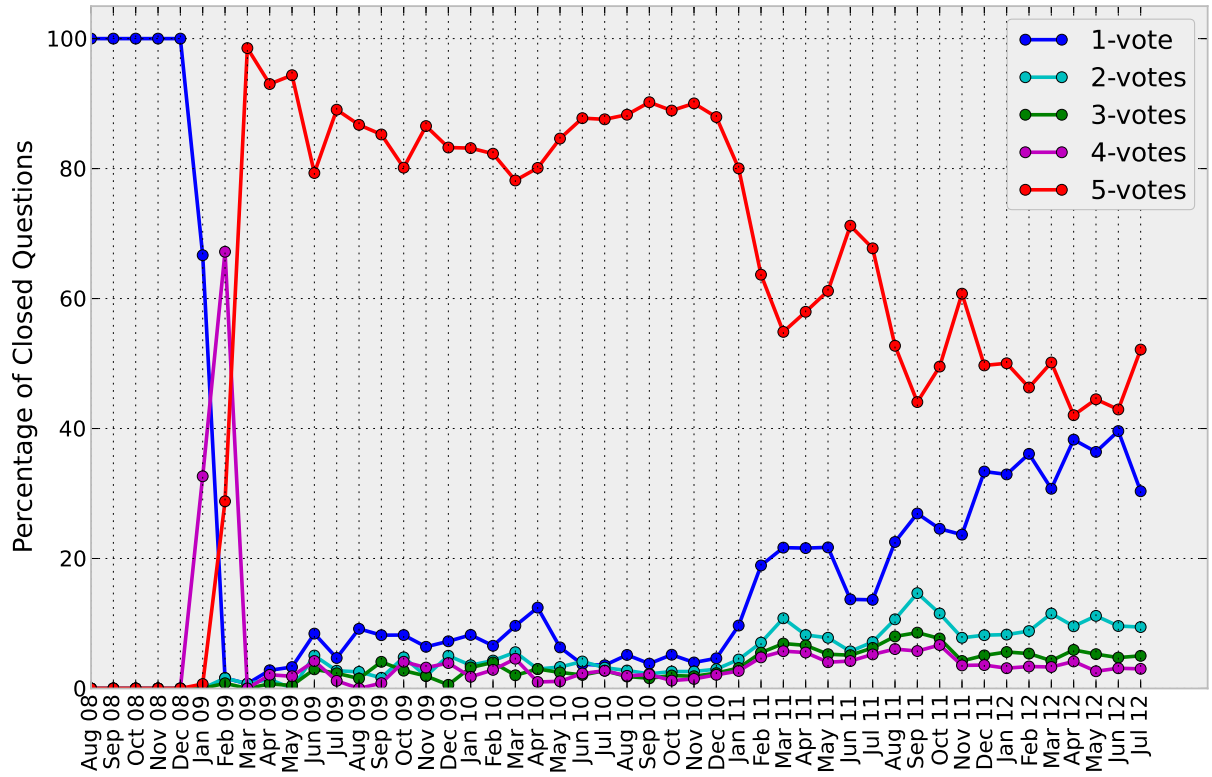


Figure 3.6: shows the temporal distribution of 'close votes' in closed questions over a 48-month period from August 2008 to August 2012. We observe that a high percentage of questions are closed due to a single ◆ moderator vote.

duplicate information to an existing question which may explain the high number. On the other hand, *Too Localized* category by definition contains questions which are programming-related but are confined to a small community and hence, the higher percentage of questions which contain source code. We see that the *Subjective* category contains the lowest percentage of questions containing source code. This could be probably because questions in this category are open-ended and invite discussions rather than an answer to a specific problem.

Figure 3.8 also shows the character length distribution of question title, body as well as the distribution of number of tags in form of a box-and-whisker plot. The top-right box plot shows that questions in the *Exact Duplicate* and *Not a Real Question* categories have lesser number of tags associated with it. The *Exact Duplicate* category may exhibit such a behavior due to user lethargy while questions belonging to the *Not a Real Question* category may be so as by definition the question marked with this label are non-programming related. A minimum of 1500 reputation points are required to create new tags on Stack Overflow [6]. The bottom left and bottom right box plots show the distribution of question title and question body lengths respectively. We do not observe a major difference in the length distributions either in title or body between categories. Both distributions are skewed i.e. there are may outliers (red points on the box plot) and the

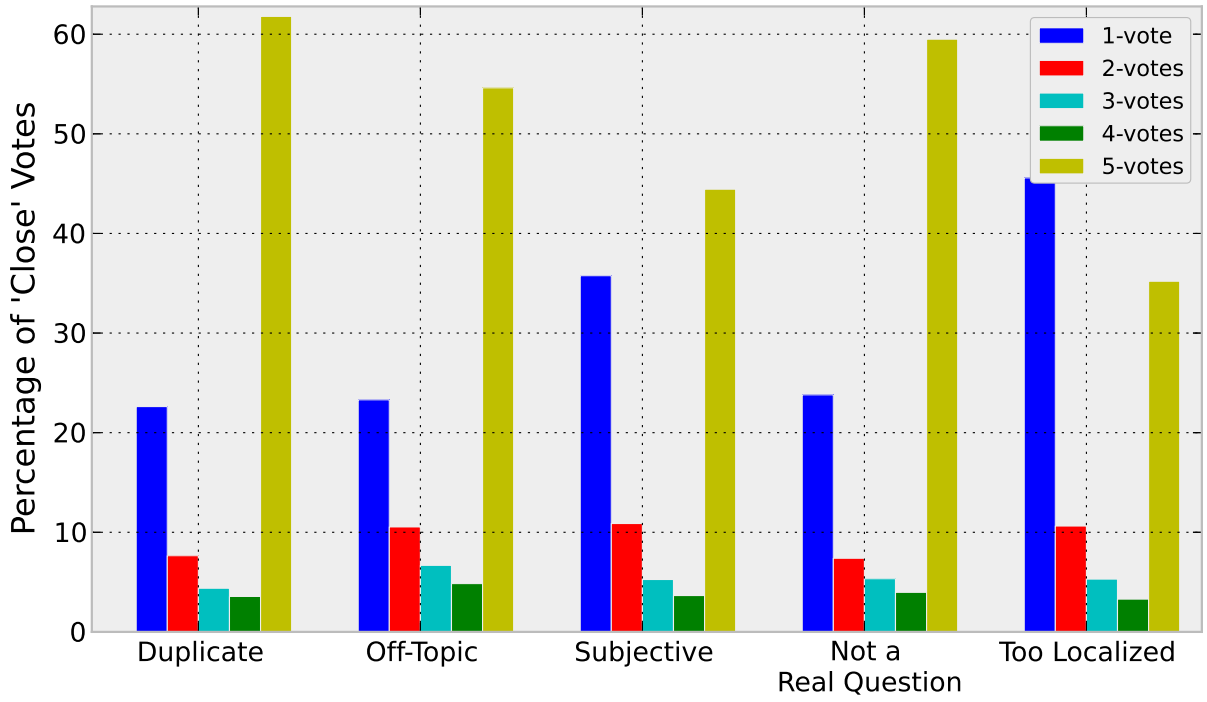


Figure 3.7: shows the 'close vote' distribution for each sub-category for all closed questions between August 2008 to August 2012. At least 1 out of 5 questions in each category are closed by a single ♦ moderator vote.

medians are approximately similar. However, in both of these distributions we once again see that the *Not a Real Question* has the lowest median value which indicates that questions belonging to this category are a clear misfit to the Stack Overflow Q&A format even in terms of content.

Question Topics

Each Stack Overflow question has some *tags* associated with it which is an identification of the topic of the question content. We analyze frequently occurring tags in 'closed' questions and bucket them into categories. Table 3.4 shows popular tags in 'closed' questions according to different categories. We see that popular tags on 'closed' questions are similar to those found overall on Stack Overflow.

We now analyze if 'closed' questions contain certain topics which are unique to their category viz. tags which relatively occur more frequently in 'closed' questions than otherwise. In order to do so, we normalize the occurrence of tags in 'closed' questions by calculating the **Normalized Tag Ratio (NTR)** for each tag. Let T_{CQ} be the set of all tags in 'closed' questions on Stack Overflow, and T_{NCQ} be the set of all tags in non-'closed' questions. We add the ϵ factor (=2.2×10^{-16}) for smoothing purposes. Then,

$$\forall t_i \in T_{CQ} \quad where \quad t_i \in \{t_1 \dots t_n\}, \quad R^i_{CQ} = \frac{count(t_i)}{\sum_{i=1}^{n} count(t_i)}$$

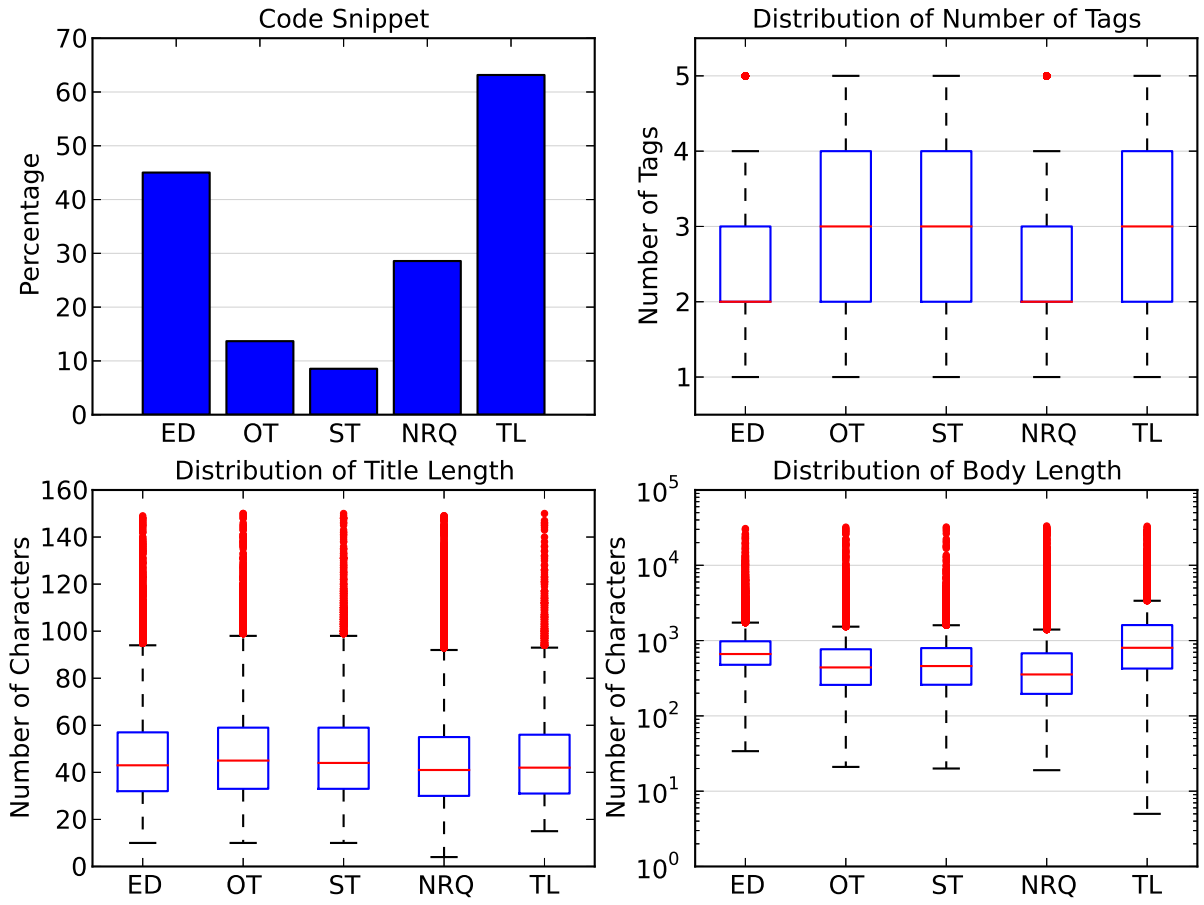


Figure 3.8: shows the percentage of code snippets in each sub-category and character length distributions of question title, body as well as the distribution of number of tags in form of a box-and-whisker plot.(ED = Exact Duplicate, OT = Off-Topic, ST = Subjective, NRQ = Not a Real Question, TL = Too Localized)

Table 3.4: Popular Tags in Closed Questions

Type	Tags
Languages	java, c++, python, c, perl, r, ...
Web2.0	php, html5, html, css, javascript, ...
Operating Systems	iOS, unix, android, osx, windows, ...
Social	Facebook, wordpress, google, ...
Miscellaneous	books, interview-questions , homework, ...

$$\forall t_j \in T_{NCQ} \quad where \quad t_j \in \{t_1 \dots t_m\}, \quad R^j_{NCQ} = \frac{count(t_j)}{\sum_{j=1}^{m} count(t_j)}$$

$$\therefore \forall t_i \in T_{CQ} \quad NTR_{t_i} = \frac{R^i_{CQ}}{R^i_{NCQ} + \epsilon}, \quad (T_{CQ} \cap T_{NCQ} \neq \emptyset)$$

Figure 3.9 shows the tags with top 30 *NTR* in closed questions on Stack Overflow. We can now see tags which are unique to 'closed' questions and find that these are quite different to the most popular tags. We notice that most tags are non-programming related; for example *working-conditions, career-development, fun* etc. We also notice that some of these tags are programming related but are on broad topics like *hidden-features, hints-and-tips* and *textbook*. These tags are usually attached to questions which require a discussion and may not focus on problem specific solutions.

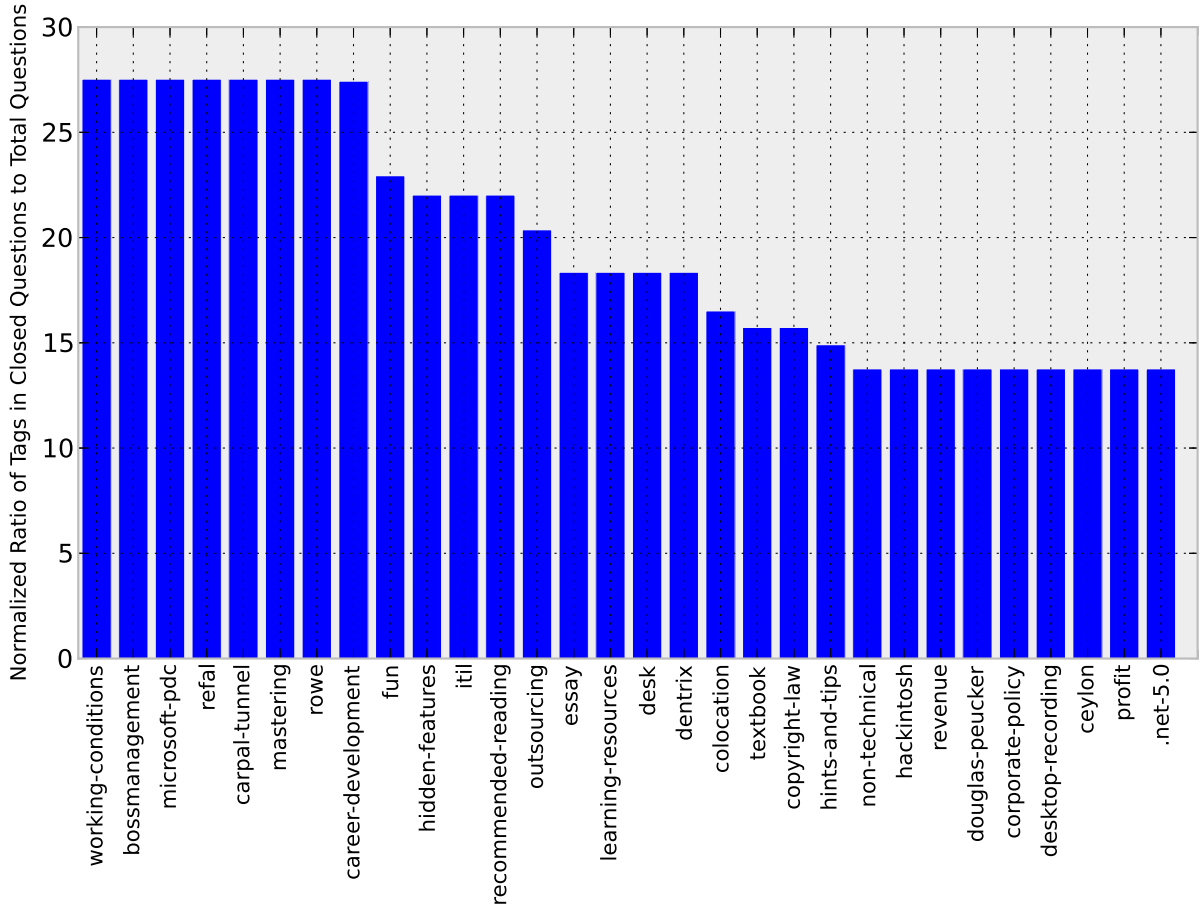


Figure 3.9: shows the tags of closed questions on Stack Overflow with top 30 Normalized Tag Ratios (NTR).

Community Value and Information Quality

A 'closed' question is irrelevant to the Q&A format and hence, implicitly suggests that the question may be low quality in context of Stack Overflow. Here, we analyze different indicators of content quality like *Favorite Votes*, *Closure Time*, *Question Scores and Answering Patterns* and *Question*

Status with respect to 'closed' questions.

Favorite Votes

Stack Overflow provides its users a feature to *favorite* a question. A *favorite vote* is an explicit statement of approval by the user that she finds the question useful and appropriate. Table 3.5 shows the cumulative distribution of 'favorite votes' on overall closed questions. The data shows that ≈ 19% of the overall 'closed' questions receive at least one *favorite vote* while ≈ 3% of those receive ≥ 5 *favorite votes*.

Table 3.5: shows the 'Favorite Vote' Cumulative Distribution for all 'Closed' questions posted between August 2008 and August 2012. Approximately 1 out of 5 'closed' questions have at least 1 'favorite' vote and 3% have at least 5 'favorite' votes.

Votes	Closed Questions
≥ 1	19,156(18.6%)
≥ 5	3,374(3.28%)
≥ 10	1,872(1.82%)
≥ 100	206(0.2%)
≥ 500	29(0.03%)
Total	102,993

However, features such as *likes* and *favorite votes* are known to be abused by users for purposes other than their intended use. Therefore, we analyze *favorite vote* distributions on different thresholds for all sub-categories of closed questions. Figure 3.10 shows the distribution of *favorite votes* at different thresholds for each category of 'closed' questions. We see that the *Subjective* category attracts a very high number of *favorite votes* from users. We perform a manual qualitative analysis on these questions and notice that the *Subjective* category contains questions like Polls, Hidden Features, Books, Tricks, Interview Questions and Open ended questions. Table 3.6 shows examples of questions in the *Subjective* category which have ≥ 100 *favorite votes*. Note that our analysis in Section 3.3 showed that *Subjective* category had the lowest percentage of questions containing code snippets. Therefore, despite the emphasis on objectivity and source code related questions by Stack Overflow guidelines we see that some amount of programming related *Subjective* questions are encouraged and appreciated by the community members.

Closure Time

We now analyze the time taken to 'close' questions on Stack Overflow. Figure 3.11 shows the closure time distribution of 'closed' question for every sub-category. The median closure times for *Exact Duplicate*, *Off Topic* and *Not a Real Question* is 6.93, 12.01 and 8.3 hours respectively. Most questions in these categories are quickly turned towards closure which may signify that their community value is relatively low than other categories. The *Subjective* and *Too Localized*

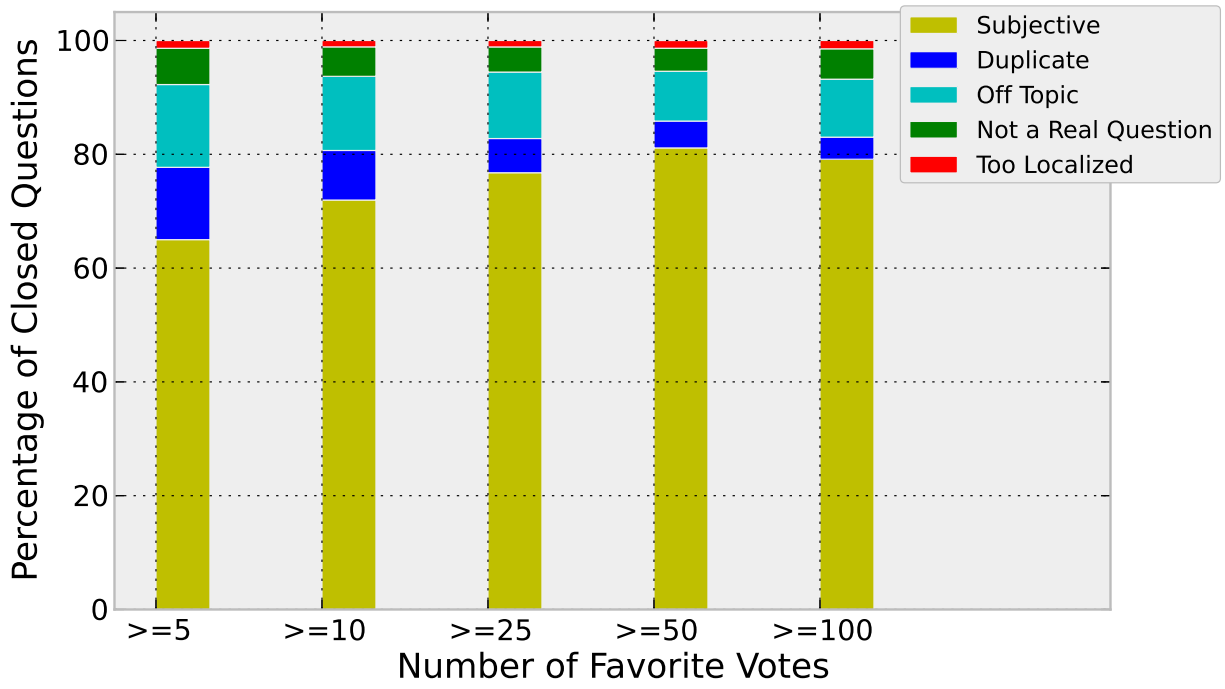


Figure 3.10: shows the distribution of 'favorite votes' on closed questions for each sub-category on various thresholds. *Subjective* category attracts very high number of *favorite votes* from users.

Table 3.6: Example questions with ≥ 100 'favorite votes' on closed questions in *subjective* category.

Favorites	Title	Answers	Views
5894	List of freely available programming books	112	569,199
2228	Hidden features of Python	100	212,589
1685	What is the best comment in source code you have ever encountered?	519	1,051,784
421	Worst security hole you've seen?	163	32,840
140	What is the most useful R trick?	34	13,197

categories have the highest median closure time ≈ 26 and 22 hours respectively. The reason for high closure time for the *Subjective* category could be because most questions (despite not being a good fit) invite discussion and opinions on broad programming related principles, guidelines, polls etc. Therefore, it takes time before these questions are answered in entirety and hence are left open for a longer time. We also notice a higher *spread* of closure times (upper quartile=586.06 days) in this category demonstrating that if a *Subjective* question is not closed within 1 day it takes a long time to close the question. The community actions indicate that these questions have not reached their maximum community value potential and hence remain open. Our prior analysis for 'close vote' distribution for *Too Localized* category in Figure 3.7 shows that the 66% of the questions in this category requires moderator intervention which may be one of the reasons for higher closure

time. Even though *Too Localized* category has a similar median closure time (22.72 hours) to the *Subjective* category very few questions require more than 6.71 days to close. This indicates that questions in the *Too Localized* category, despite similar median closure time, reach their maximum community value potential relatively earlier than those in *Subjective* category. We also find that each category contains some outliers i.e. each category contains some questions which take a long time to be marked as 'closed'. Table 3.7 shows the close vote distribution pattern on questions with outlier closure times in each category.

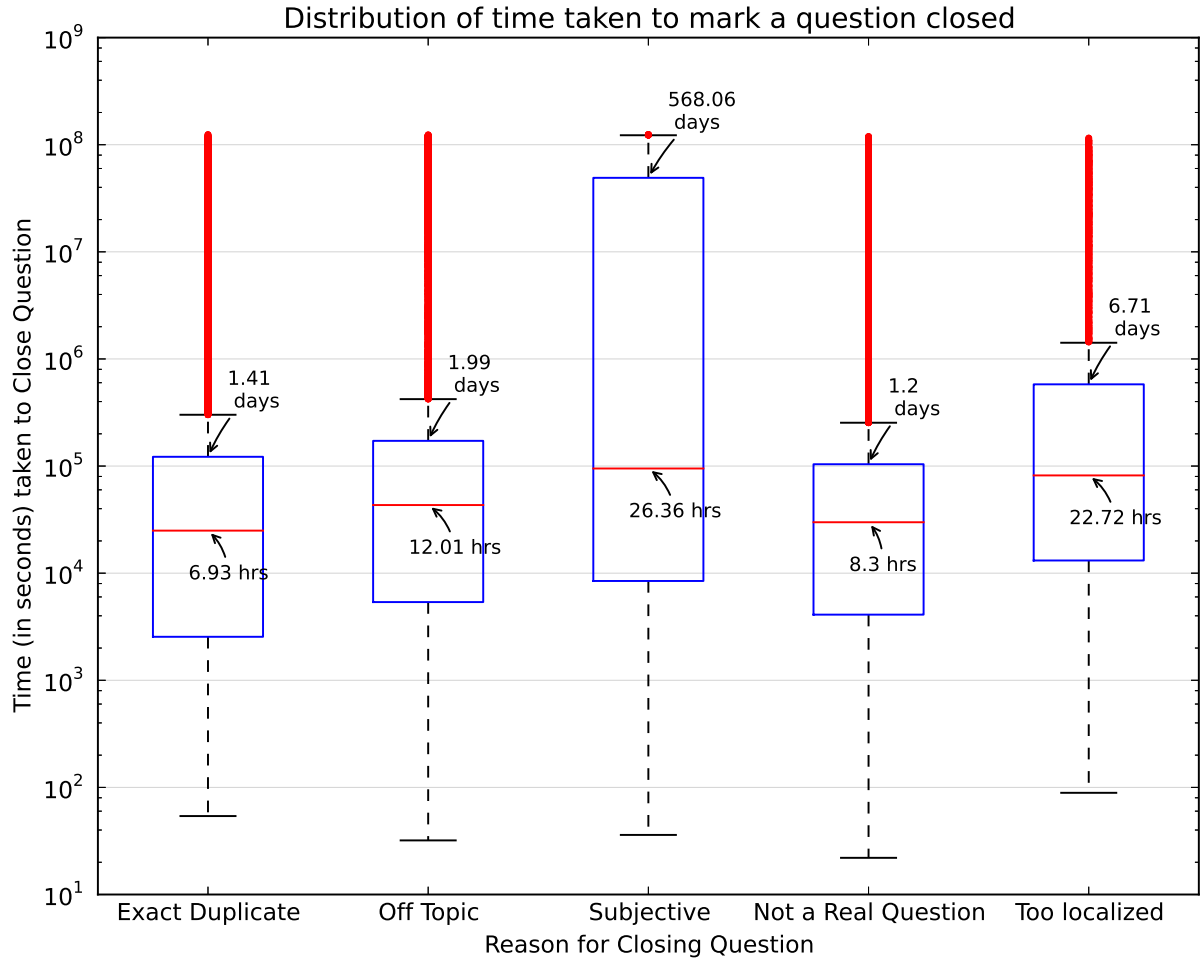


Figure 3.11: shows the distribution of time taken to close questions for each category in the form of a box-and-whisker plot.

We see that all the outlier questions have a very high percentage of ◆ moderator intervention on question closure time. This indicates that these questions are indeed outliers in terms of content too as the community prefers to keep these questions open to reach their maximum community value potential.

Table 3.7: Number of Close Votes on outliers from each category

Category	1-vote	2-vote	3-vote	4-vote	5-vote
Duplicate	**55.44%**	11.68%	4.25%	2.18%	26.45%
Off-Topic	**42.06%**	16.21%	6.31%	3.47%	31.96%
Subjective	**64.64%**	16.66%	4.9%	2.26%	11.54%
Not a Real Question	**46.97%**	9.52%	6.28%	3.5%	33.74%
Too Localized	**68.22%**	11.85%	3.62%	1.86%	14.45%

Question Scores and Answer Patterns

Figure 3.12 shows various question scores and answer patterns on 'closed' questions in Stack Overflow. We first look into the percentage of answers (PA), percentage of accepted answers (PAA)[2] and percentage of accepted answers given an answer (PAC) on each category of 'closed' questions on Stack Overflow. We see that a large percentage of 'closed' questions receive answers from users. The *Duplicate* and *Subjective* categories also have a relatively higher PA and PAA than other categories. The higher PA and PAA on *Duplicate* questions suggest that despite the fact that the question content is an exact duplicate of others the community is eager to answer the question. Such behavior may also be exhibited by answerers to garner more reputation points in the form of answer votes and *accepted* answers. This may also explain why we see a very low PAA in the *Not a Real Question* category as users are smart enough to pick questions which have a higher probability of receiving up votes. Recall that our earlier analysis reveals that questions belonging to the *Not a Real Question* category are low in information content quality. We also analyze question score patterns on each category of 'closed' questions on Stack Overflow. We calculate percentage of questions with negative score (QN), percentage of questions with ≥ 5 score (QT) and percentage of questions with zero score (QZ). We find that *Not a Real Question* has the highest QN and once again indicates that questions in this category are very low in quality. We observe a similar pattern for *Too Localized* category and may indicate that the community in general frowns upon questions which are too confined to certain sections of the programming fraternity. We see that *Subjective* category has a very high QT and this falls in line with our earlier hypothesis that questions in this category despite being not fit to the website are immensely popular and therefore, draws large number of votes. We see similar QZ values (between 30–50%) on all categories of 'closed' questions which demonstrates that some questions do not get any approval from the community. We would like to comment that we do not observe any familiar statistical distributions like power-law on any of these question scores and answer patterns.

Question Status Apart from being marked as 'closed', a Stack Overflow question can also be given a *locked*, *community wiki* and *protected* label. Table 3.8 shows the distribution of 'closed' questions with a *locked*, *community wiki* and *protected* label. A *locked* question can not receive any new answers or any form of votes on question-answers. A question is primarily *locked* by

[2]A question asker can mark an answer *accepted* if the answer solves the problem faced by the question asker. Accepted answers lead to gain in reputation points to answerers.

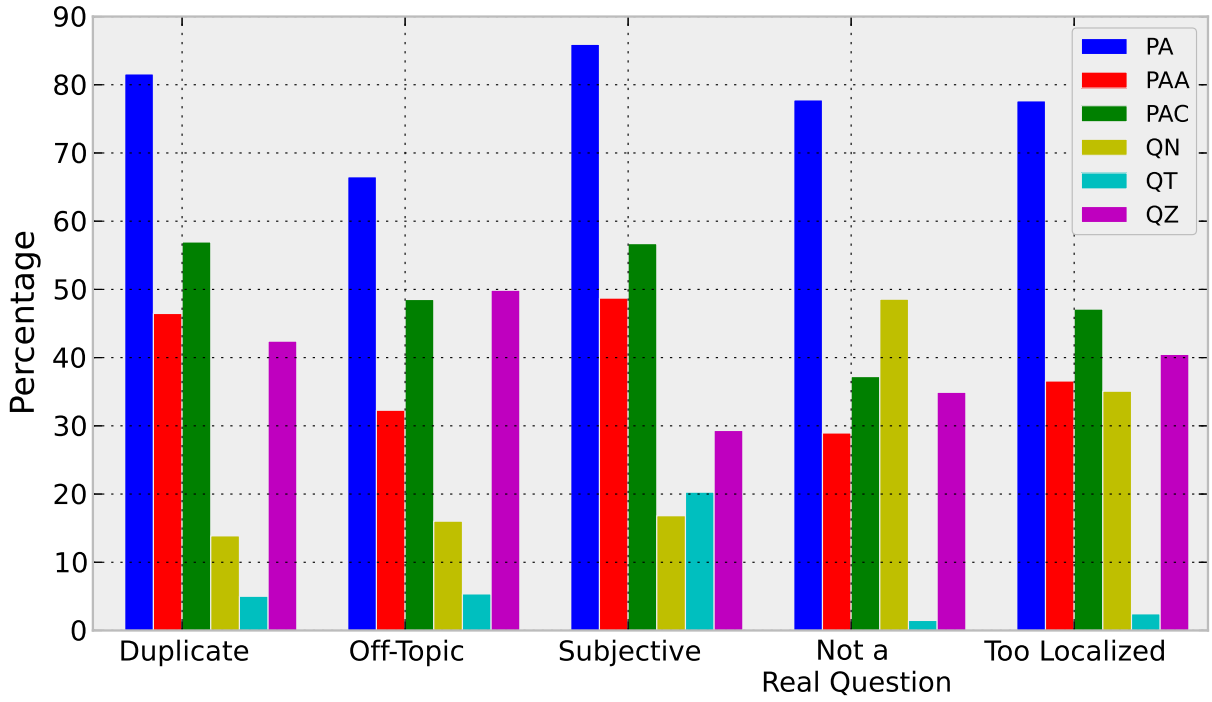


Figure 3.12: shows the question scores and answering patterns of users on closed questions in each category. PA = Percentage of Answers, PAA = Percentage of Accepted Answers, PAC = Percentage of Accepted Answers given that a 'closed' question has an answer, QN = Percentage of Questions with Negative Score, QT = Percentage of Questions with ≥ 't' Score (t=5), QZ = Percentage of Questions with Zero Score.

◆ moderator to prevent gaming or abuse of the system by users to garner reputation points [18]. We observe that *Exact Duplicate* and *Off Topic* categories are most prone to reputation gaming and therefore, marked as *locked*. A *community wiki* label is an intent to 'donate' and transfers ownership of the question from the asker to the community. The goal of Stack Overflow is to be a knowledge base of programming information and therefore *community wiki* posts play a significant role in achieving that goal [17]. We see that questions from the *Subjective* category contain a high number of *community wiki* donations. We hypothesize that this would be due to the nature of *subjective* questions as these contain discussions, opinions on programming topics which may be "never ending" (philosophical rather than factual). A *protected* label is an intent to prevent noisy answers like "Thank You", "+1" from new users who may not understand the guidelines of the forum. A *protected* label prevents newly registered users from answering these question [20]. Once again we see that a high percentage of questions from *Subjective* category are marked as *protected*. This demonstrates that *Subjective* questions are very attractive and "fun" questions to users although they may not fit into the Stack Overflow guidelines.

Table 3.8: shows the distribution of 'closed' Questions in the Stack Overflow with labels *locked, community wiki* and *protected*.

	Number of 'Closed' Questions		
Category	**Locked**	**Community Wiki**	**Protected**
Exact Duplicate	732(33.8%)	160(9.9%)	36(10.3%)
Off Topic	**1180(54.5%)**	273(16.8%)	70(20.1%)
Subjective	188(8.7%)	**978(60.3%)**	**202(58%)**
Not Real Question	50(2.3%)	192(11.8%)	28(8%)
Too Localized	114(0.6%)	10(0.6%)	12(3.4%)
Total	2,264	1,613	348

Characterization Summary

We now summarize key findings from our characterization study.

- We see an increasing trend in the percentage of 'closed' questions over time – in particular *Exact Duplicate* and *Not a Real Question* – with a steep rise after January 2011
- We find a positive correlation with a high confidence value between new registered users and the percentage of 'closed' questions
- We observe a decrease in community participation to mark a question as 'closed' over time which has probably led to increase in work load for ♦ moderators
- Popular tags on 'closed' questions are very similar to overall questions but tags unique to 'Closed' questions are vague and non-programming related
- Questions from the *Subjective* category do not follow the Q&A format but are very popular and have high community value. They also take relatively longer time to be marked as 'closed'. Questions from the *Not a Real Question* category take least amount of time to be closed and are low in community value.
- Despite a very high percentage of presence of source code, questions in *Too Localized* are not very popular in the community
- *Exact Duplicate* and *Off Topic* questions are relatively more attractive to reputation gamers

Broader Implications

Our characterization study on Stack Overflow also reveals broader implications for content quality on other CQA websites. The intricate procedures like voting, moderation etc. laid down by the Stack Overflow community helps keep low quality content to the minimum (we recall that only 3% questions are marked 'closed'). These procedures can be an important data point for other CQA websites to adopt and adapt. We observe that most low quality content on Stack Overflow is posted

by newly registered users. Therefore, an early feedback mechanism to newly registered users can help with their website interaction and encourage them to post better quality content. We see that the strict adherence to guidelines by the community members leads to content being flagged for moderation despite being popular. Our analysis of 'closed' questions marked as *subjective* shows that popularity of content is not a defining factor of quality. Therefore, adherence to guidelines (and not content popularity) is a significant factor for content quality maintenance.

3.4 'Closed' Question Prediction

In the second part of our study, we build a predictive model to automatically detect a 'closed' question on Stack Overflow. We formulate the prediction of 'closed' questions on Stack Overflow as a binary classification task.

Features for Classification

We investigate **18** features based on *User Profile*, *Community Process*, *Question Content* and *Textual Style* for our prediction task. Table 3.9 shows different categories of feature sets used by our system for 'closed' question prediction. *User Profile* features are based on user's participation activity while *Community Process* features are based on Stack Overflow community contributions in the form of votes, accepted answers etc. *Question Content* features are calculated by extracting the content from questions and *Textual Style* features characterizes the writing and posting style of the question asker. It is important to note that there may be other distinguishing features for 'closed' questions (for example - answering patterns) but the aim of the study is to predict a 'closed' question at its *creation time*. Hence, we can not make user of these features for our predictive model. The reputation of the user at question creation time is an excellent feature by intuition however, this data is not made available by Stack Overflow. Therefore, we use *Community Process* features to offset for this missing data. In addition, questions are routinely edited (title, body and tags) by experienced community users. However, there is no mechanism to get the original text of the question. All these factors make prediction of a 'closed' question difficult and challenging.

While most of the features are self-explanatory, below we explain some of the higher order features below (calculated at time of question creation) –

Badge Score (BS):
Let $\{b_1 \dots b_n\}$ be the badges earned by the user, then

$$BS = \sum_{i=1}^{n} \frac{1}{\#\text{users who have } b_i}$$

Post Score (PS):
Let $\{q_1 \dots q_n\}$ be the set of previous questions asked by the user and $\{a_1 \dots a_m\}$ be the set of previous answers posted by the user, then

Table 3.9: shows the different categories of feature sets used for 'closed' question prediction

Set	Category	Number	Features
A	**User Profile**	3	*Age of Account* *Badge Score* *Previous Posts with Negative Score*
B	**Community**	3	*Post Score* *Accepted Answer Score* *Favorite Score*
C	**Question Content**	3	*Number of URLs* *Number of Stack Overflow URLs* *Number of Popular Tags*
D	**Textual Style**	9	*Title Length* *Body Length* *Number of Tags* *Number of Punctuation Marks* *Number of Short Words* *Code Snippet Length* *Number of Special Characters* *Number of Lower Case Characters* *Number of Upper Case Characters*

$$PS = \sum_{i=1}^{n} score(q_i) + \sum_{j=1}^{m} score(a_i)$$

Favorite Score (FS):
Let $\{fq_1 \dots fq_n\}$ be the set of questions asked by the user which have been favourited and $\{fa_1 \dots fa_m\}$ be the set of answers posted by the user which have been favourited, then

$$FS = \sum_{i=1}^{n} score(fq_i) + \sum_{j=1}^{m} score(fa_i)$$

Accepted Answer Score (AAS):
Let $\{aa_1 \dots aa_n\}$ be the set of answers posted by the user which have been accepted. We give an individual score of 15 to each accepted answer, therefore

$$AAS = \sum_{i=1}^{n} 15$$

Number of Popular Tags (#PT):
Let $T = \{t_1 \dots t_n\}$ be the tags present in the question, and $PT = \{pt_1 \dots pt_m\}$ our pre-derived set of

popular tags on Stack Overflow[3], then

$$\#PT = \|T \cap PT\|$$

Experimental Testbed, Setup and Classification

Stack Overflow contains 102,993 'closed' questions between August 2008 to August 2012. Out of these questions, 1302 questions do not have any information about the question asker. We ignore these questions and consider the remaining **101,691** 'closed' questions as our positive class. The percentage of non-'closed' questions (negative class) is very high (97%) than 'closed' questions (3%) and therefore, leads to the formation of an imbalanced dataset. Learning with imbalanced data is a research challenge and has attracted wide spread attention of researchers in the machine learning community. Various approaches have been proposed in literature to address the nature of imbalanced datasets. One such approach is to randomly under-sample the majority class data or over-sample the minority class data to make the dataset balanced [70]. In order, to make our dataset balanced we under-sample the majority class (non-'closed' questions or -ve class) and draw **101,691** random samples. However, random sampling may result in sample bias and lead to loss of information. In order to eliminate this sample bias, we perform under-sampling by drawing several random independent subsets from the majority class (-ve class) and training multiple classifiers based on each of these subsets along with the minority class (+ve class). We then evaluate our classifier across these multiple data instances and report our results. In our experiments, we draw 10 independent random subsets from **101,691** samples from the non-'closed' questions (negative majority class) and train 10 classifiers based on each of these 10 subsets along with **101,691** samples from 'closed' questions (positive minority class). Therefore, in total we have **203,382** data samples across both classes for each classification run.

Figure 3.13 outlines the cumulative distribution function (CDF) plots for four features – Age of Account, Code Snippet Length, Post Score and Body Length – in our experimental testbed for 'closed' and non-'closed' questions. The CDF plots strongly indicate that the features chosen for our classification task are weak discriminators. An ensemble framework gives the ability to combine the power of such weak discriminators to form a strong predictive model. Previous approaches in information and question quality prediction on CQA services have also observed good classification performance with ensemble learners like Stochastic Gradient Boosted Trees (SGBT) [27, 87]. In addition, we experimented with various classification algorithms including Support Vector Machines, Naive Bayes, Logistic Regression etc. and find that the *Stochastic Gradient Boosted Trees* gives the best performance. Stochastic Gradient Boosted Trees (SGBT) is an ensemble learning technique which combines information from *weak* predictive models (primarily built on decision trees) to form a *strong* classifier [59]. The stochastic approach randomly sub-samples the training data without replacement before the construction of each tree and hence, avoids over fitting on the data. Table 3.10 provides a summary of our testbed and experimental setup.

[3]We obtain popular tags by calculating tag distribution of all tags in our dataset.

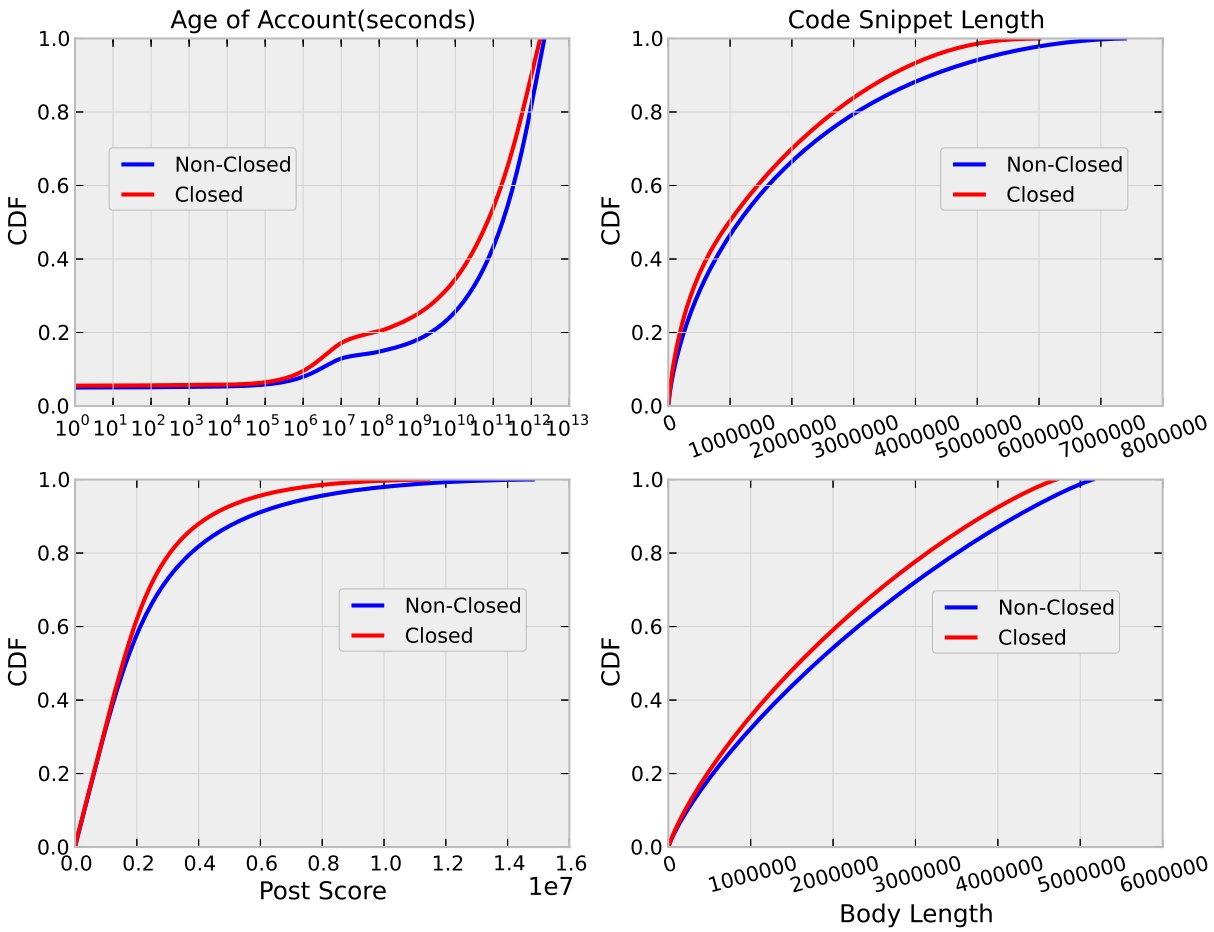


Figure 3.13: shows the cumulative distribution plot of four selected features used in our classification task. We see that all features are weak discriminators.

We choose the most efficient parameters for learning rate (0.1) and sub-sample size (0.7) for SGBT. We use a 70-30% training-testing split and perform 10-fold cross validation on each classification run of our positive class versus random sample of negative class.

Classification Results and Evaluation

Table 3.11 shows the confusion matrix for our classification experiments. We are able to accurately classify 69.6% of ‘closed ’ questions and 70.9% of non-‘closed’ questions.

Our characterization study indicates that there is no intuitive heuristic or metric to predict a ‘closed’ question. Hence, in order to understand the effect of features to predict ‘closed’ questions, we incrementally add feature sets to our classifier and record the performance. We use three standard information retrieval metrics – F1 score, Accuracy and Area Under the ROC curve (AUC) to evaluate our classifier. Figure 3.14 shows the performance of our classifier on Accuracy, F1 score

Table 3.10: Details of Experimental Setup

Dataset	203,382 questions
'Closed' (+ve class)	101,691
Non-'Closed' (-ve class)	101,691 (10-times) random sample with replacement
Classifier	Stochastic Gradient Boosted Trees
Learning Rate	0.1
Sub-sample size	0.7
Classification Runs	10 (for each +ve/-ve pair)
Feature Sets	{A}, {A, B}, {A, B, C}, {A, B, C, D}
Train-Test Split	70%-30%
Cross Validation	10-folds

Table 3.11: Confusion Matrix – Classification Results

		Predicted	
		Closed	**Non-Closed**
True	**Closed**	**69.6%**	30.4 %
	Non-Closed	29.1%	**70.9%**

and AUC metrics when feature sets are incrementally added. We see that each feature set has a positive effect on the performance of the classifier across all metrics. This suggests that the all our feature sets are important for prediction.

Feature Importance

One of the advantages of using SGBT is that it outputs a list of important features used for classification. Figure 3.15 shows the most important features for classification. Overall, we see that almost all features contribute towards our prediction model. The top five features for classification are – *Punctuation Marks*, *Special Characters*, *Code Snippet Length*, *Age of Account* and *Short Words*. We analyze the distribution of all the top five features. We observe a higher presence of *Punctuation Marks* and *Special Characters* in non-'closed' questions indicating that non-'closed' questions are more descriptive than 'closed' questions. The distribution of *Code Snippet Length* as shown in Figure 3.13 shows that code snippets (if present) are longer in non-'closed' questions than 'closed. This shows that code samples posted on non-'closed' questions are comparatively more detailed. The distribution of *Age of Account* in Figure 3.13 confirms that a newly registered user is more prone to post a 'closed' question. Overall, we see that 'closed' questions are less informative and descriptive than non-'closed' questions.

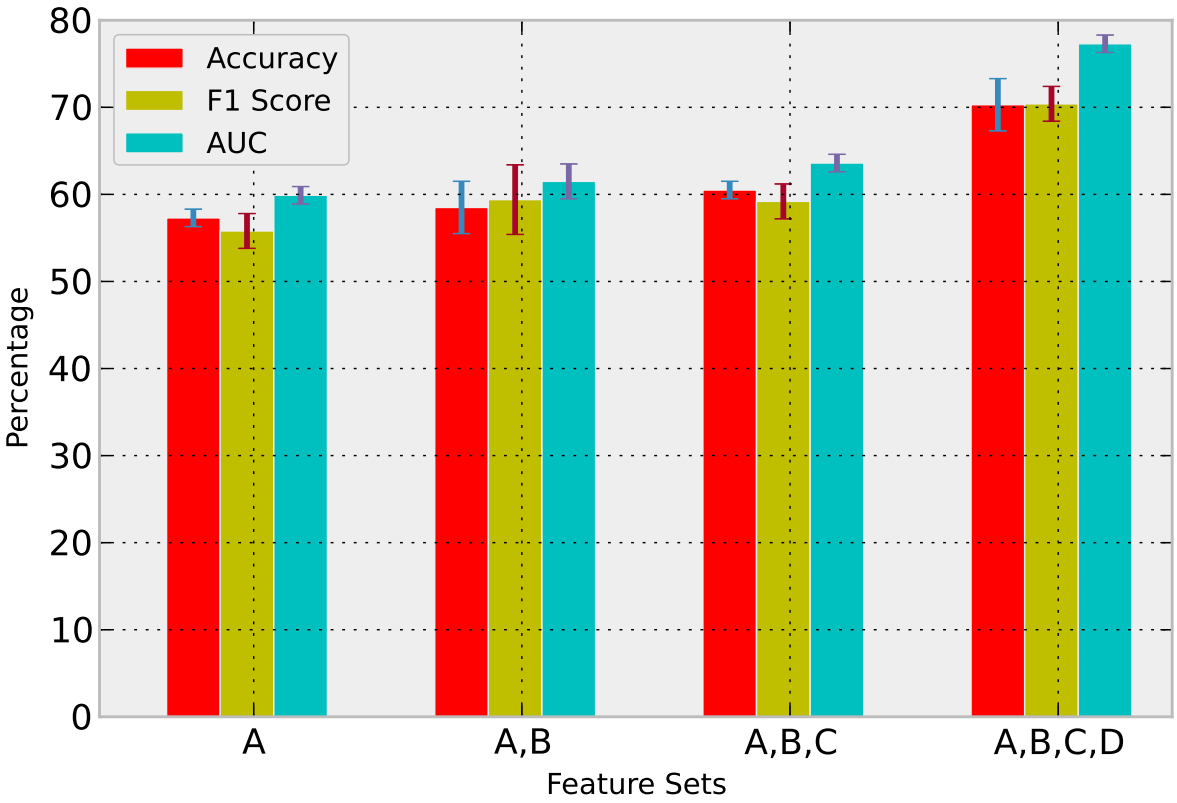


Figure 3.14: shows classifier performance with Accuracy, F1 and Area Under the ROC curve (AUC) metrics when feature sets are incrementally added. Note the incremental improvement in performance of our classifier on every feature set addition.

3.5 Conclusion

Stack Overflow is an extremely popular programming Community Question Answer (CQA) website for developers throughout the world. Stack Overflow uses a *karma* based incentive system to maintain the quality of content on its website. However, despite these guidelines users post questions which do not fit Stack Overflow's Q&A format. Questions which are deemed unfit for Stack Overflow are marked as 'closed' by experienced users and community moderators. We present the first study of 'closed' question on 4 years of publicly available data from Stack Overflow. We divide our study into two phases – In the first phase, we conduct a characterization of 'closed' questions posted between August 2008 to August 2012. Our characterization reveals that *subjective* 'closed' questions are popular and high quality while *not a real question* are low in quality. We also notice decrease in community participation on question closure over time and find that *Duplicate* and *Off Topic* are more prone to reputation gaming. The analysis reveals some broader implications on content quality maintenance for CQA websites. In the second phase, we construct a predictive model for identifying a 'closed' question using an ensemble learning technique

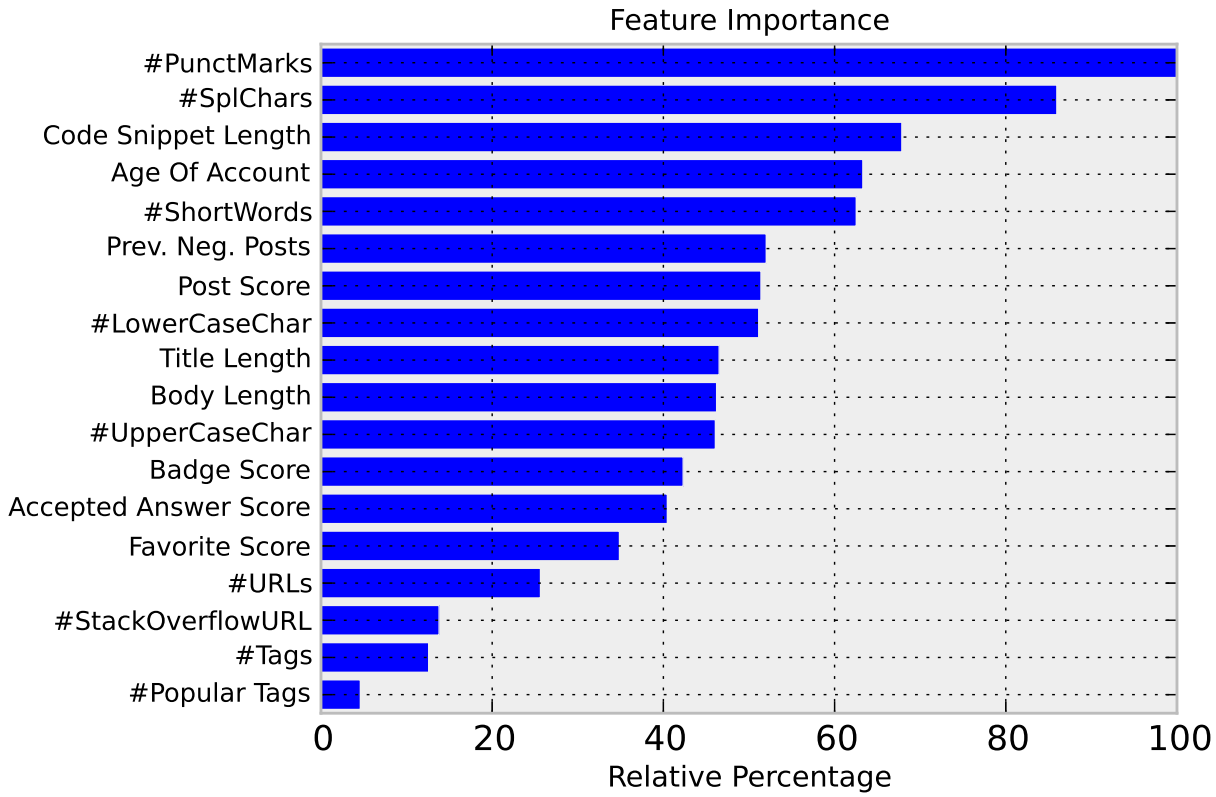


Figure 3.15: shows the relative feature importance of all 18 features in our predictive model.

and report 70.3% accurate predictions overall. Feature analysis reveals that ‘closed’ questions are relatively less informative and descriptive than non-‘closed’ questions.

Chapter 4

Deleted Questions on StackOverflow

In the previous chapter, we looked at *closed* questions on Stackoverflow which were low quality. However, some questions on Stackoverflow are deemed unwanted and hence are *deleted* from the website. In this chapter, we perform and in-depth study of *deleted* questions on Stackoverflow.

4.1 Introduction

Research Motivation and Aim

There has been an increase in the presence of Community Based Question Answering (CQA) website services like Yahoo! Answers, Ask.com and Quora on the Internet. Stack Exchange is a growing network of thematic question-answering websites with each website dedicated to a specific field of expertise [75]. It consists of 107 CQA websites with 4.1M users, 7.3M questions, 13.2 M answers and 8.5M visits per day.[1] CQA websites on Stack Exchange span across different orthogonal themes like *Technology* (Web Applications, Game Development), *Culture* (Travel, Christianity), *Arts* (Photograph, Scientific Fiction) and *Sciences* (Mathematics, Physics). Stack Overflow is a programming based CQA and the most popular Stack Exchange website consisting of 5.1M questions, 9.4M answers and 2.05M registered users on its website.[2]

Stack Overflow has detailed, explicit guidelines on posting questions and it maintains a firm emphasis on following a question-answer format. The community strongly discourages questions which could generate chit-chat, opinions, polls etc. and employs elected moderators to ensure content quality maintenance. Stack Overflow is a free, open (no registration required) website to all users on the Internet and hence, it is a necessity to maintain quality of content on the website [27]. Stack Overflow is driven by the goal to be an exhaustive knowledge base on programming related topics and hence, the community would like to ensure minimal possible *noise* on the website. However, despite of the presence of question posting guidelines and an ebullient moderation community, a significant percentage of questions on Stack Overflow are extremely poor in nature.

[1] `https://stackexchange.com/about`

[2] `http://stackoverflow.com/`

Questions are a fundamental aspect of any CQA website and the presence of poor quality content may affect user experience. Prior work also shows that poor quality content on a CQA website may drive users away from the platform and adversely affect the traffic [93]. Therefore, it is important to study poor quality questions and develop mechanisms to minimize them on the website. In this work, we focus our attention on *deleted* questions (poor quality) on Stack Overflow. Questions which are very poor in quality or extremely off topic in nature are deleted from the website. Table 4.1 shows examples of deleted questions on Stack Overflow. We see that most of these questions are very poor in quality and of little worth to the community.

Table 4.1: shows examples of deleted questions on Stack Overflow

Id	Title	Score
1644242	Get drive information (free space, etc.) for drives on Windows and populate a memo box	-50
2022741	plzz can u help me	-9
14771464	NDTV iPad app screen design	-8
2351964	I need to see this q fast ..pleash	-8
3077356	PostgreSQL stupidity	-8
2984348	hi question about mathematics	-3

Our research aim is to understand the phenomena of *deleted* questions on Stack Overflow to gain insights about the nature of poor quality questions. In addition, we also develop a predictive framework to detect the probability of a question to be deleted at the time of question creation. Such a predictive framework would help the moderators of the Stack Overflow community to detect a potential *deleted* question on Stack Overflow. However, a deleted question on Stack Overflow is an explicit feedback to the question asker that her question does not conform to the guidelines. A predictive framework can convey immediate feedback to the question asker about her question. This may help her to revise her question and improve it in order to avoid deletion. Therefore, prediction of a deleted question at post creation time has two distinct benefits – (i) An immediate feedback mechanism to the question asker and (ii) A complementary mechanism to aid community moderators in their daily moderation tasks.

Research Contributions

We perform the first large scale study on poor quality or *deleted* questions on Stack Overflow. We make the following research contributions -

- We analyze deleted questions on Stack Overflow posted over ≈5 years and conduct a characterization study. We perform a longitudinal study of deleted questions, community voting patterns, deletion behavior by question owners and discover question quality structure. We make our data on *deleted questions* publicly available for research purposes.

- We develop a predictive model using a machine learning framework to detect a deleted question at the time of question creation. We experiment with 47 features based on four diverse categories and report 66% accurate predictions. We also perform analysis of our feature space and report features with best discriminatory powers.

4.2 Deleted Questions on Stack Overflow

In this section, we briefly discuss about deleted questions on Stack Overflow. Figure 4.1 summarizes various details about deleted questions on Stack Overflow.

Why are questions deleted?

The goal of Stack Overflow is to be the most extensive knowledge base of programming related topics. Hence, it would like to keep the *noise* on their website as low as possible. Therefore, questions on Stack Overflow which are extremely off topic or very poor in quality are deleted from the website [12]. In addition, questions which are dormant viz. have no activity over a significant period of time are also deleted. Also, 'closed' questions (questions which are deemed unfit for Stack Overflow) which do not serve as useful sign points may also be deleted. The *Why* block of Figure 4.1 corresponds to this section.

Who can delete a question?

Experienced users with 10,000+ reputation points can cast 'delete votes' in order to delete a question. Question authors can delete their own questions if – (i) the question has zero answers OR (ii) the question has only one answer but no up votes. However, a community elected moderators (super users of the website) can delete any questions at their discretion [15] The *Who* block of Figure 4.1 corresponds to this section.

How are questions deleted?

The number of votes required to delete a question scales to the number of answers and the up votes on the answers of those questions. This notwithstanding, a minimum of 3-votes and a maximum of 10-votes are required to delete a question. It must be noted that a deleted question is not physically deleted from the website but *soft deleted*. Moderators, developers and experienced users (10000+ reputation) points are able to view these questions. However, deleted questions do not appear in search results [11] The *How* block of Figure 4.1 corresponds to this section.

What happens after a question is deleted?

Once a question is deleted there are two possibilities – (i) it remains deleted and (ii) it is *undeleted*. The procedure for *undeleting* questions is similar to that of deleting a question. The *What* block of Figure 4.1 corresponds to this section.

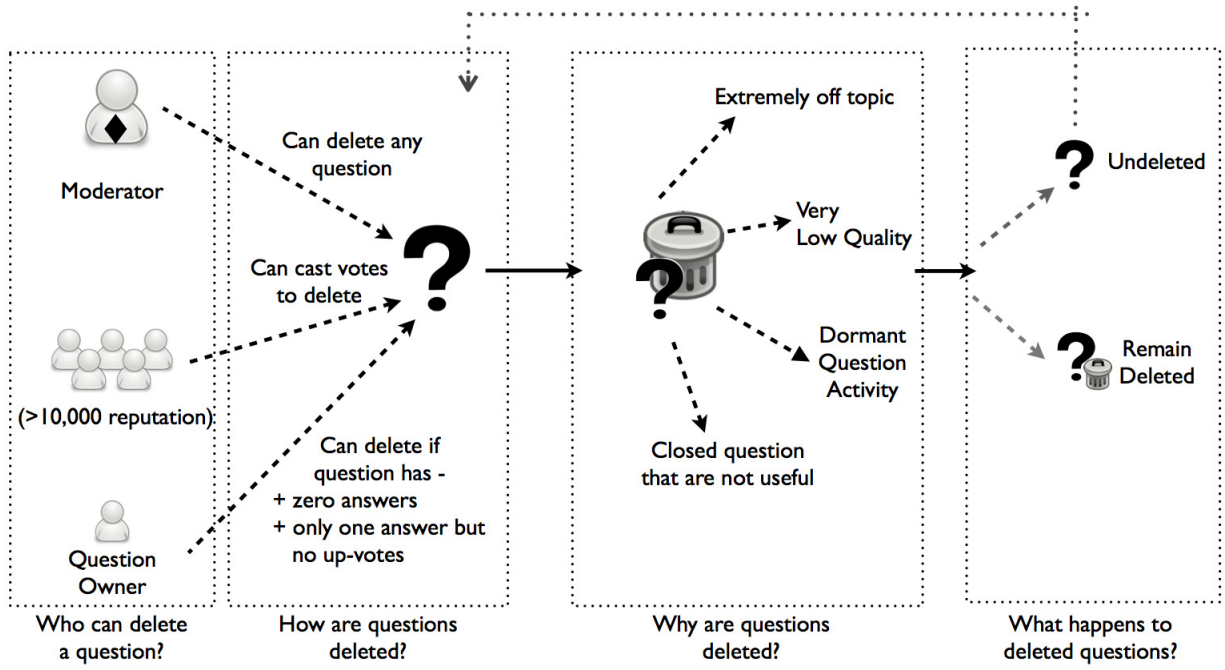


Figure 4.1: shows the details about procedures and community guidelines to delete a question on Stack Overflow.

4.3 Characterization of Deleted Questions

In this section, we present our findings on deleted questions on Stack Overflow.

Dataset Description

Stack Overflow provides a periodic database dump of all user-generated content under the *Creative Commons Attribute-ShareAlike* [34]. The database dump contains publicly available information of questions, answers, comments, votes and badges from the genesis of Stack Overflow (August 2008) to the release time of the dump. Table 4.2 shows the months in which Stack Overflow provided database dumps between August 2008 to June 2013.

We download all the available 24 database dumps for our study. Table 4.3 shows the overall statistics of user-generated content on Stack Overflow between August 2008 (inception) to June 2013 (current). The statistics show that Stack Overflow is a very popular programming CQA with 5.1M questions, 9.4M answers and 2.05M registered users.

However, the database dumps provided by Stack Overflow do not directly contain information about deleted questions. Hence, we analyze the entire 24 database snapshots over $\approx$ 5 years to construct our dataset. Concretely, questions available in the earlier database snapshots (August 2009 – March 2013) but absent in the most recent snapshot (June 2013) are *deleted* from Stack Overflow. We obtain **293,289** ($\approx$0.29M) deleted questions between September 2009 and June 2013

Table 4.2: shows the months in which Stack Overflow provided database dumps between August 2008 to June 2013.

Months	2009	2010	2011	2012	2013
January		♦	♦		
February		♦			
March		♦			♦
April		♦	♦	♦	
May		♦			
June			♦		♦
July		♦			
August	♦	♦		♦	
September	♦	♦	♦		–
October	♦	♦			–
November	♦	♦			–
December	♦		♦		–
Total	24 Database Snapshots				
Information	Questions, Answers, Comments, Votes and Badges				

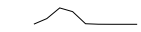
Table 4.3: overall statistics of user-generated content on Stack Overflow between August 2008 – June 2013

Users	2.05M (936k askers, 630k answerers)
Questions	5.1M (60.22% with accepted answers)
Answers	9.4M (32.67% marked as accepted)
Votes	42.3M (70.5% +ve, 6.7% favorites)
Ratio of Answers to Questions	1.84

by following this procedure. However, we notice a small (but sizeable) percentage of questions in this set which are not deleted but wrongly captured. We eliminate such errors by inspecting HTML pages and obtain a final experimental dataset of **270,604** (≈0.27M) deleted questions. Table 4.4 shows descriptive statistics of user-generated content of deleted questions on Stack Overflow between August 2008 to June 2013. We would like to point out that our experimental dataset contains a subset of all deleted questions on Stack Overflow. This limitation is due to the sporadic sharing of database dumps by Stack Overflow (and not due to the procedure we use to find deleted questions). Questions deleted between two data snapshots would not be captured in our experimental dataset. However, it must be noted that our dataset contains the maximum possible deleted questions which can be obtained given the publicly available Stack Overflow database snapshots. We also make our experimental dataset publicly available for research purposes under the *Creative Commons Attribute-ShareAlike*.[3]

[3] `http://correa.in/datasets.html`

Table 4.4: shows statistics of user-generated content of deleted questions on Stack Overflow between August 2008 to June 2013. The distribution sparklines begin at minimum and end at their corresponding maximum value (log-scale). The distributions are generated using Gaussian kernel density estimates.

	Mean	Median	Min	Max	Distribution
Questions per year	54,120	49,221	17,514	102,623	
Answers	2.96	1.0	0.0	637.0	
Score	0.15	2.22×10^{-16}	-56.0	695.0	
Views	221.65	80.0	1.0	296,466	
Favorites	3.59	1.0	0.0	1530.0	
Comments	2.26	1.0	0.0	77.0	

Increase in Deleted Questions Over Time

We now perform a temporal trend analysis of deleted questions on Stack Overflow. Figure 4.2 shows the ratio of deleted questions to total questions in each month over a 49-month period between September 2009 and June 2013. We would like to recollect that the first database snapshot provided by Stack Overflow is on August 2009 and hence, we do not have information of deleted questions prior to this snapshot. We observe that on an average ≈8% of questions are deleted on Stack Overflow. We also notice an anomalous increase in percentage of deleted questions ≈15% after the month of May (2010 2011). We posit that these abrupt rises could be due to periodic *Deletion Question Audits* conducted by the Stack Overflow community around the month of May [19].

Figure 4.3 shows the cumulative distribution area chart of deleted questions over a 49-month period between September 2009 and June 2013. The area chart depicts that there is a sharp increase in the total number of deleted questions over time (even with an average of ≈8%). We notice a particularly steep increase after May 2011. Therefore, despite the presence of comprehensible and explicit question posting guidelines – Stack Overflow receives a high number of extremely poor quality questions which are not fit to exist on its website. Hence, it is important to perform a longitudinal study about deleted questions on Stack Overflow.

Community Takes Long Time to Detect but Swift Action by Moderators

Stack Overflow delineates an elaborate procedure to delete a question. We recall that experienced community members viz. users with 10,000+ reputation points can cast 'delete votes' to delete a question. We analyze these 'delete vote' patterns to gain insights into the community participation dynamics on poor quality questions on Stack Overflow. Hence in this section, we restrict our analysis to **62,949** deleted questions which have received 'deleted votes'. Figure 4.4 shows the distribution of time taken to receive the first 'delete vote' on a deleted question on Stack Overflow. We see that ≈80% questions take at least 1 month(or more) to receive its first 'delete vote' while

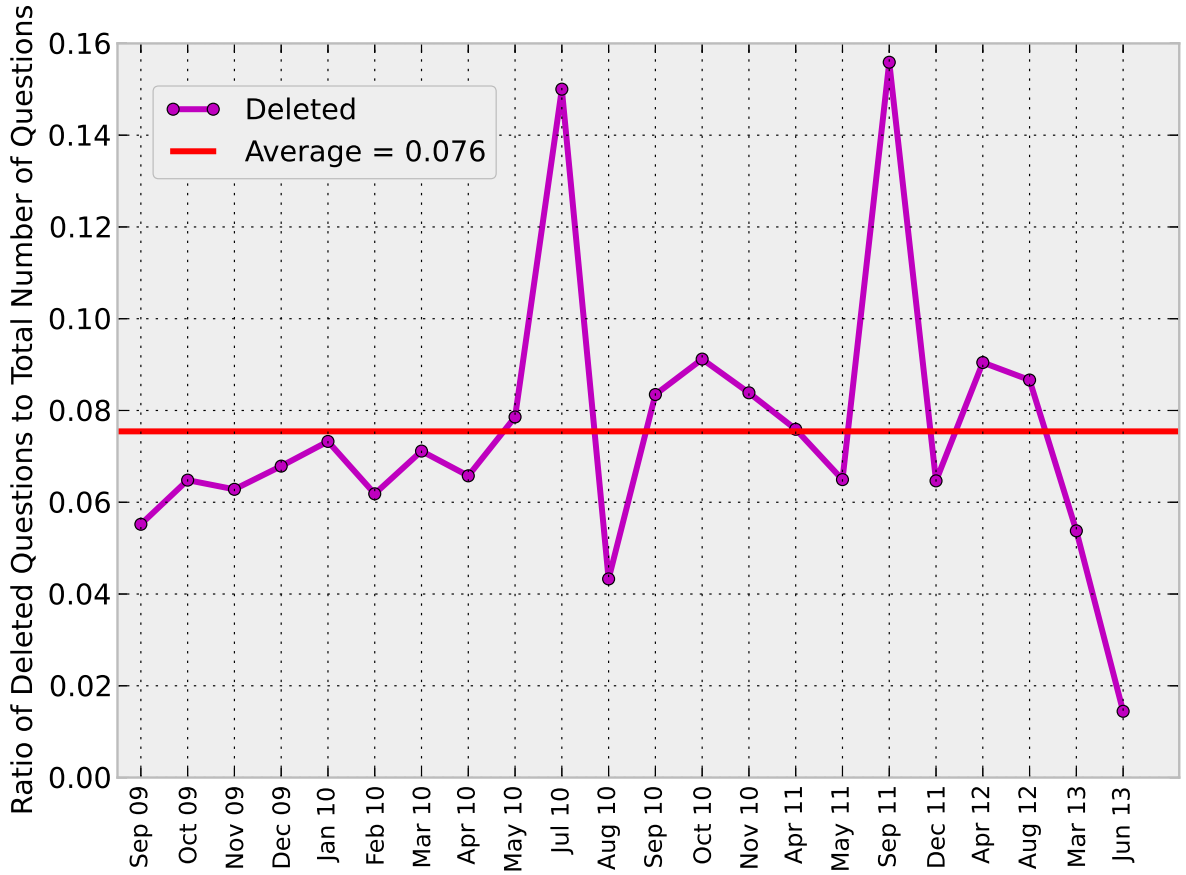


Figure 4.2: shows the ratio of deleted questions to total questions in each month over a 49-month period between September 2009 and June 2013.

≈50% of the questions take 6 months(or more). Overall, 50 percentile of the questions take >164 days to receive their first 'delete vote'. Therefore, majority of deleted questions take a significant amount of time to receive its first delete vote.

We now analyze the distribution of 'deleted votes' in our dataset. Table 4.5 shows the 'deleted vote' distribution on deleted questions posted and Figure 4.5 shows the temporal distribution of 'delete votes' on deleted questions. We observe that ≈80% questions receive 1 'delete vote' and ≈14% questions receive 3 'delete votes' before they are deleted. This shows that once a question receives the required number of 'delete vote'(s), moderators move quickly to take appropriate action. The moderators are driven by the Stack Overflow motto to keep a low signal-to-noise ratio in order to maintain high content quality. We also note that ≈23% of questions in our experimental dataset have received 'delete votes' i.e. questions which are voted to be deleted by experienced users. Therefore, 75% of questions in our dataset are never voted for deletion by the community. This reveals that the major duty to detect extremely poor quality questions is on the encumbrance of the moderator. This, coupled with the 'delete vote' pattern-action behavior (relatively long time

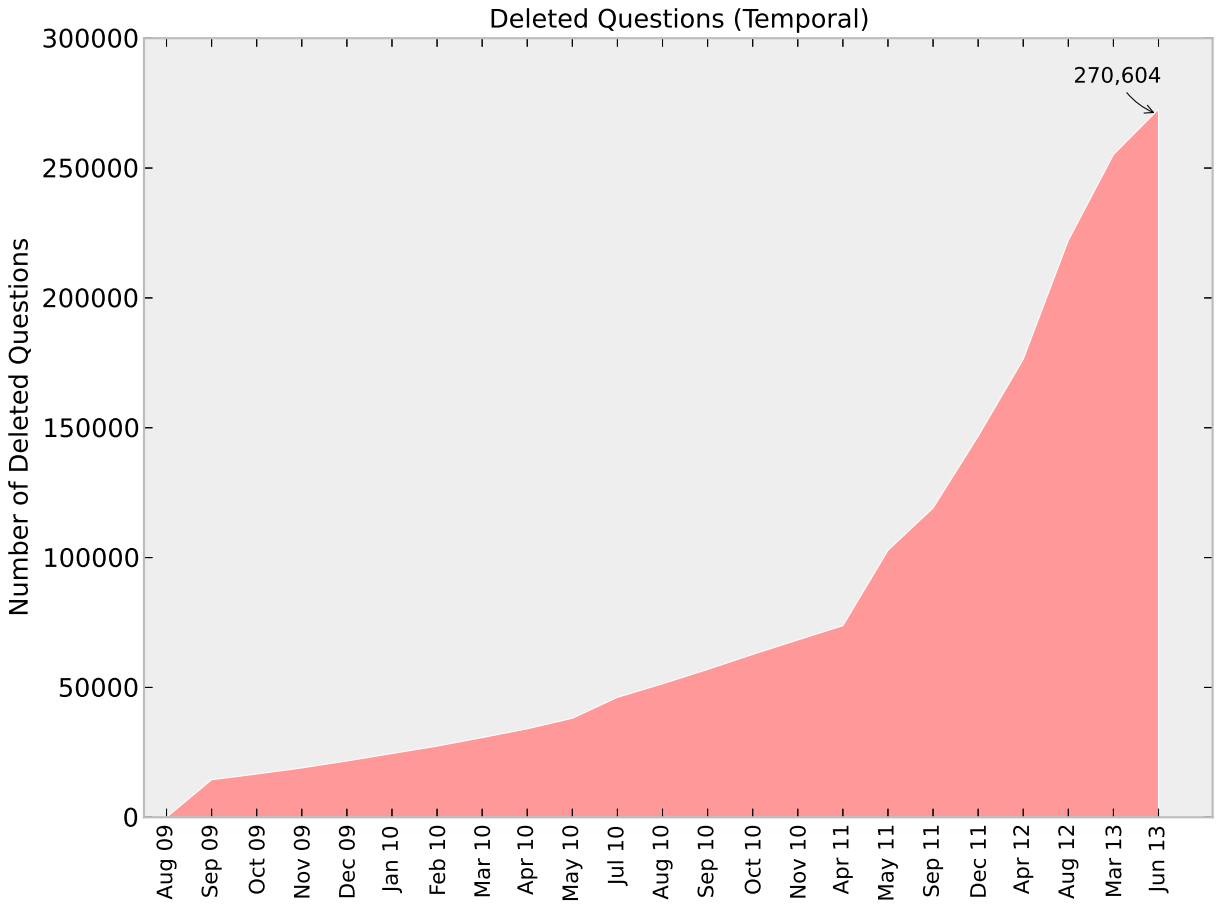


Figure 4.3: shows the cumulative distribution of deleted questions over a 49-month period between September 2009 and June 2013. We notice a sharp rise in the number of deleted questions over time.

but swift action) motivates the impending need to have a predictive system for the benefit of the moderators on Stack Overflow.

Table 4.5: shows the 'deleted vote' distribution on deleted questions posted between June 2009 and June 2013.

Votes	Deleted Questions
1-vote	50,012 (79.45%)
2-votes	2,736 (4.35 %)
3-votes	9,009 (14.3%)
4-votes	463 (0.74%)
5+-votes	729 (1.16%)
Total	62,949

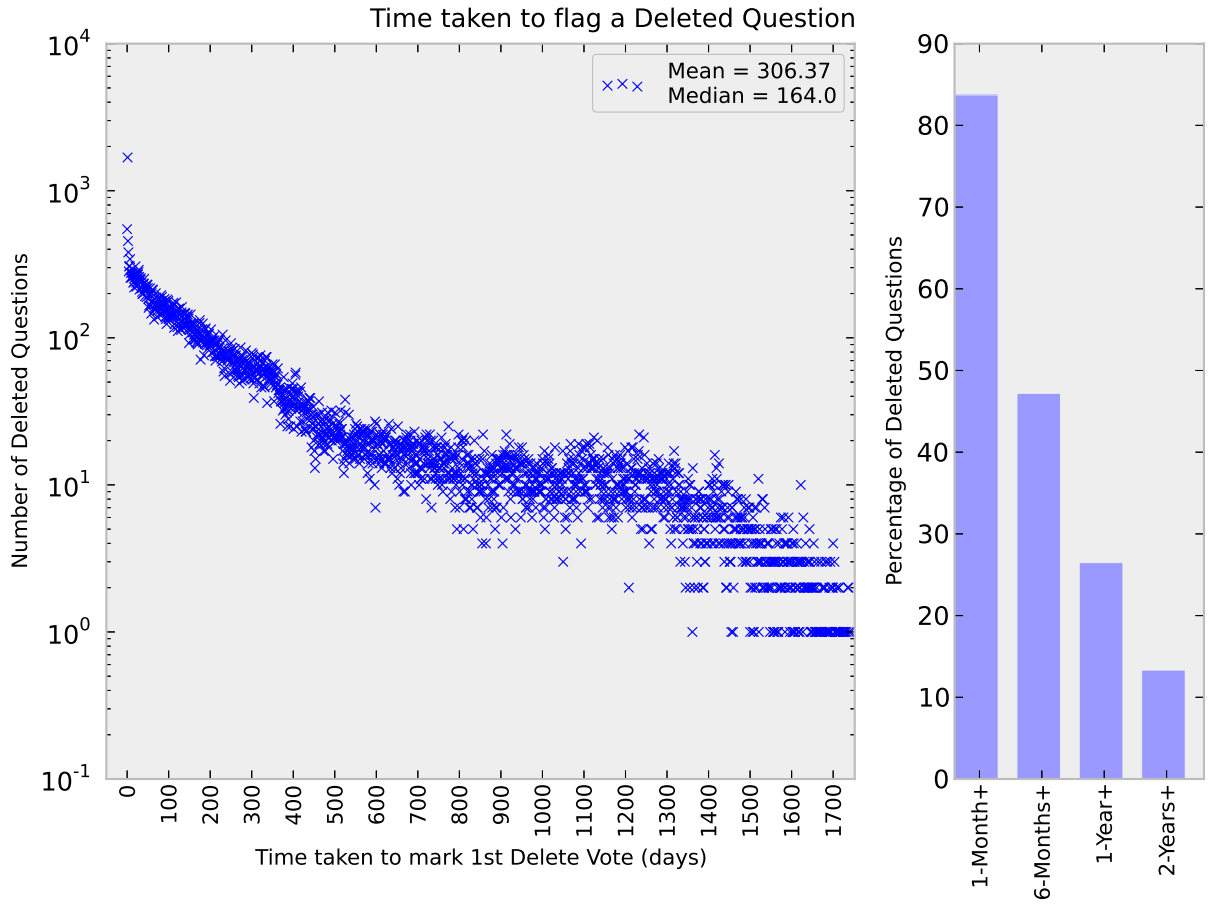


Figure 4.4: shows the distribution of time taken to receive the first delete vote on a deleted question on Stack Overflow.

Authors Delete Questions to Salvage Reputation

We recall that a question on Stack Overflow can either be deleted by the author of the question or by a moderator. However, this information is not directly available in the publicly available data dumps provide by Stack Overflow. We also recall that questions on Stack Overflow are not digitally deleted i.e. they are hidden from the site and do not appear in search results. We notice that this information about question deletion initiator (question author or moderator) is available on the unique web address of the question. Figure 4.6 shows the web page screenshots of – (i) question deleted by moderator (left) and (ii) question deleted by author (right).

We download the unique web pages of deleted questions in our experimental dataset and employ a regular expression to extract this information. Figure 4.7 shows the distribution of question deletion initiator (moderator or author) on Stack Overflow. We notice that ≈87% (237,163) of questions are deleted by a moderator while ≈12% (33,330) or 1 out of 8 questions are deleted by the question author. A negligible percentage (0.04%) of questions (111) do not belong to either

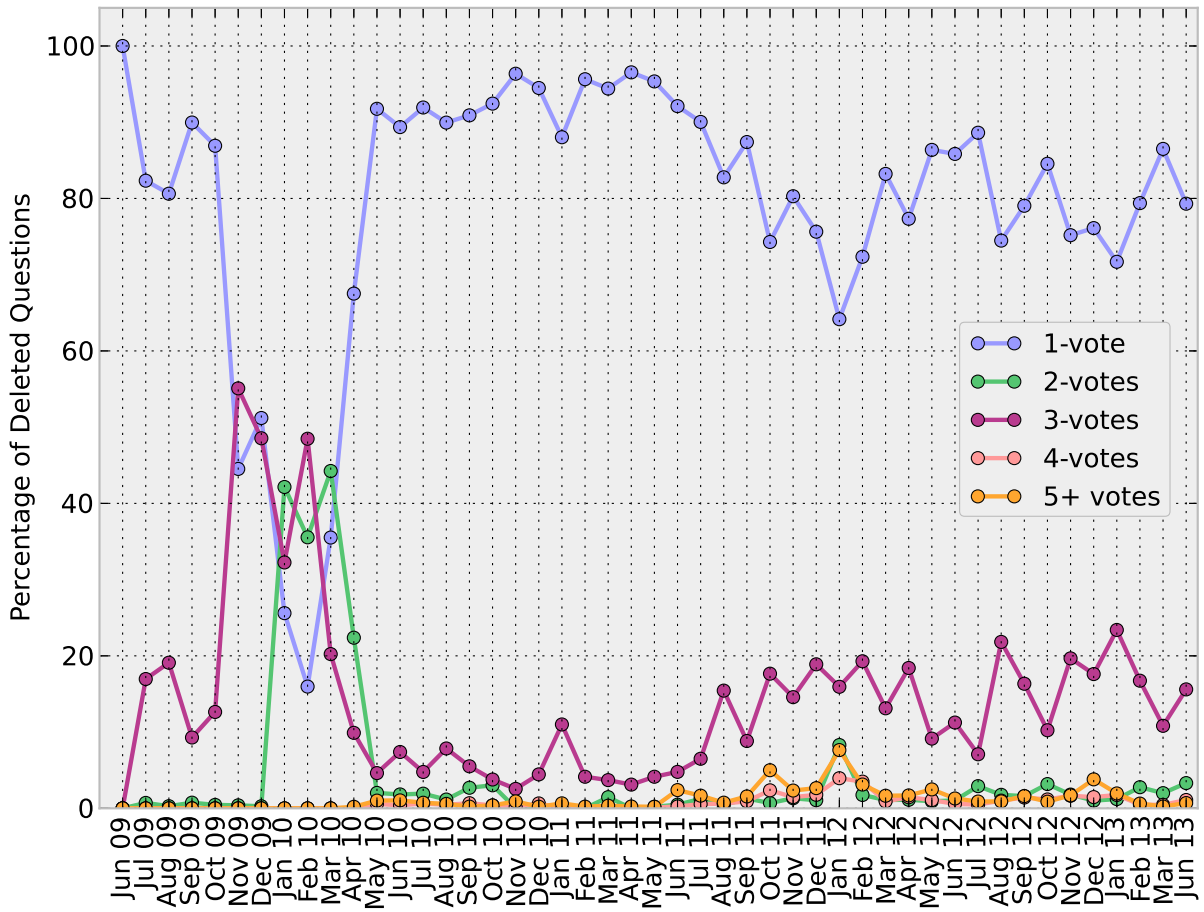


Figure 4.5: shows the temporal distribution of 'delete votes' on deleted questions in Stack Overflow between June 2009 and June 2013.

category. This may be a bug in the data dump and hence, we ignore these questions in this section.

We now analyze multiple question attributes based on the question deletion initiator (author or moderator) and make observations. Figure 4.8 (Top-Bottom, Left-Right) shows the distributions of – (i) time taken to delete a question (box-and-whisker plot), (ii) account age of question owner (box-and-whisker plot), (iii) posts prior to deleted question creation (percentile plot) and (iv) deleted question score (percentile plot) for author and moderator deleted questions. The Top-Left box-and-whisker plot reveals that the distribution of time taken to delete a question by the question author is relatively lower than when done by a moderator. The Top-Right box-and-whisker plot shows that the question owner age of account is relatively higher of author deleted questions than moderator deleted questions. This points out that question owners of author-deleted questions are more experienced on the website(in terms of time) than question owners of moderator-deleted questions. The Bottom-Left percentile plot shows that prior posts – posts made by the question owner prior to the time of deleted question creation – of author-deleted questions is higher than

Figure 4.6: shows the web page screenshots of two questions deleted by moderator (left) and author (right) on Stack Overflow.

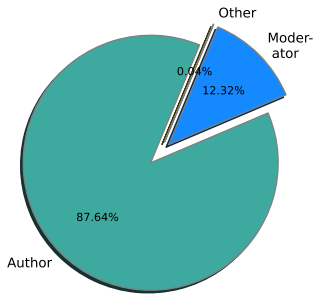


Figure 4.7: shows the distribution of question deletion initiator (moderator or author) on Stack Overflow

those deleted by a moderator. The Bottom-Right percentile show shows that question scores of author-deleted questions are higher than those of moderator-deleted questions at very low (<20) and negative question scores.

The above observations show that despite being less experienced on the website, question owners of author-deleted questions have more prior posts and have higher question scores on deleted questions than those of moderator-deleted questions. We argue that question owners of author-deleted questions exhibit such a behavior as they want to maintain a healthy reputation earned on Stack Overflow [43]. The owners of author-deleted questions observe that their questions are attracting down votes which affects their *reputation*. In order to stem further decrease reputation points, question owners see deletion of question as a quick fix and therefore, proceed to delete the posted question in an attempt to salvage their reputation.

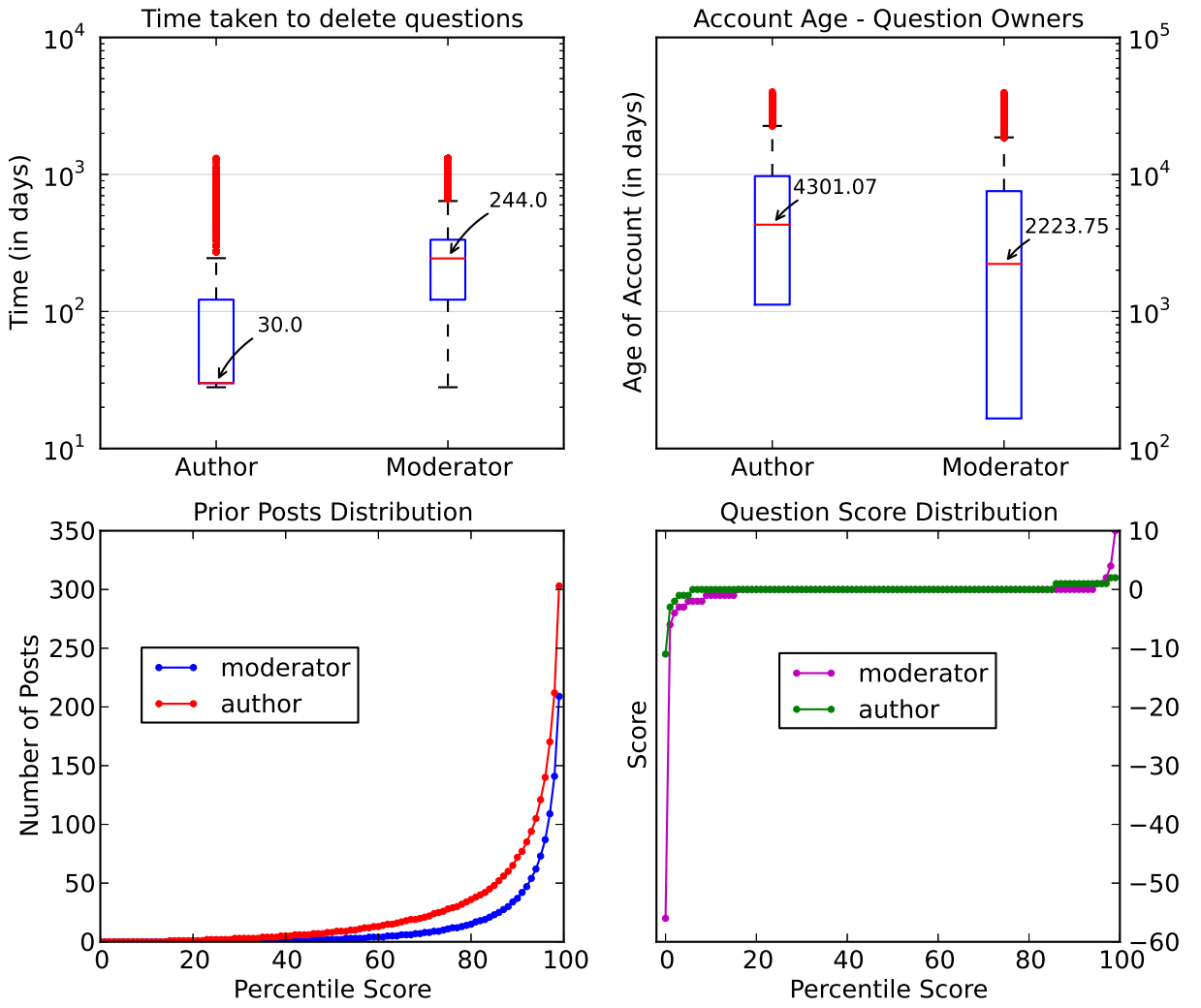


Figure 4.8: shows the distributions of – (top-left) time taken to delete a question via box-and-whisker plot, (top-right) account age of question owner via box-and-whisker plot, (bottom-left) posts prior to deleted question creation via percentile plot and (bottom-right) deleted question score via percentile plot for author and moderator deleted questions.

Question Quality Pyramidal Structure

Questions on Stack Overflow are marked 'closed' if they are deemed unfit for the question-answer format on Stack Overflow and indicate low quality. A question can be marked as 'closed' due to five reasons – *duplicate*, *subjective*, *off topic*, *too localized* or *not a real question* [53]. We find that there are **254,446** (0.25M) 'closed' questions on Stack Overflow between August 2008 June 2013. In this section, in addition to our experimental dataset we utilize this data to analyze question quality indicators for deleted and 'closed' questions. We also make observations about the relative quality of deleted questions in context to 'closed' questions on Stack Overflow.

Community Value

Figure 4.9 shows various quantities of question quality indicators for 'closed' and deleted questions on Stack Overflow. We see that deleted questions have higher percentage of questions with zero *score* than 'closed' questions. A question *score* is the net worth of the usefulness of a question as determined by the Stack Overflow community. ≈80% of deleted questions have a zero *score* which indicates that most deleted questions are of very little worth to the community. We also observe that in comparison to 'closed' questions – deleted questions have a lower percentage of question with greater than zero *score*, *favorite* votes and *view counts*. A *favorite* vote is equivalent to an explicit expression of interest or subscription on a question. Only ≈5-6% of deleted questions attract a positive question *score* or *favorite* votes. This indicates that deleted questions are generally of very little worth and interest to the Stack Overflow community. In addition, ≈10% of deleted questions have >0 *view counts* which is twice as lower than that of closed questions (≈23%). We also find that deleted questions have lesser percentage of code snippets but slightly higher number of characters in their body than 'closed' questions. Nonetheless, we observe that deleted questions fare inferior on most indicators in comparison to 'closed' questions indicating very poor quality.

Question Topics

Stack Overflow questions contain user supplied *tags* which indicate the topic of the question. We analyze the tag distribution of closed and deleted questions and compare them to the overall tag distribution on Stack Overflow. There are a total of **36,643** tags on all questions in Stack Overflow. Figure 4.10 shows the venn diagram of tag distributions of questions on Stack Overflow.

We see that tags on 'closed' questions are a subset of the overall tags which occur in regular questions. In contrast, an appreciable number of tags on deleted questions (≈10%) are not found either in 'closed' or regular questions. We extract such tags found in deleted questions for further analysis. The topics of 'closed' questions are relevant to the community despite the questions themselves being unfit for the Stack Overflow format. However, some of the topics on deleted questions are extremely off topic to the interests of the community. Figure 4.11 shows the word cloud of the top-50 tags that occur on deleted questions. We find a very high presence of low quality tags like *homework*, *job-hunting* and *polls* on deleted questions. These topics also show that deleted questions are of extremely poor quality.

Effort to Improve Question Quality

The goal of Stack Overflow is to be a comprehensive knowledge base for programming related topics. Therefore, a question on Stack Overflow may be edited by privileged users (experienced community members and moderators) to maintain content quality. A question on Stack Overflow has three major sections which can be edited – title, body and tags. In addition, users without sufficient privileges can *suggest edits* to questions. These suggestions are then reviewed by privileged users and are brought into effect at their discretion. In this part, we analyze edit histories of deleted questions on Stack Overflow. Table 4.6 shows four edit types (edit tags, edit body, edit title and suggested edits) for deleted questions in our experimental dataset. We see that ≈93% of deleted

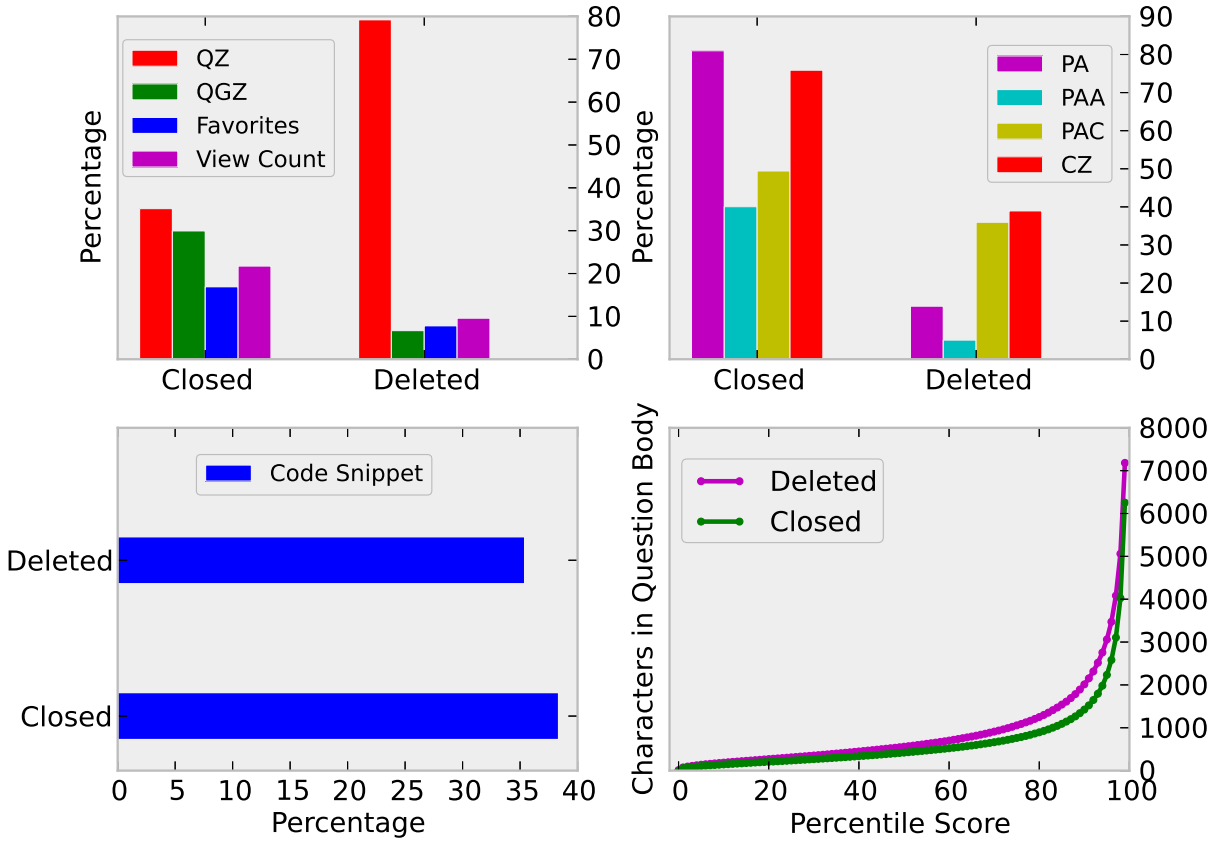


Figure 4.9: shows the quantities of question quality indicators for 'closed' and deleted questions on Stack Overflow.(QZ=Percentage of Questions with Zero Score, QGZ=Percentage of Questions with Greater than Zero Score, PA=Percentage of Answers, PAA=Percentage of Accepted Answers, PAC=Percentage of Accepted Answers given that a question has an answer, CZ=Percentage of Comments with Zero Scores)

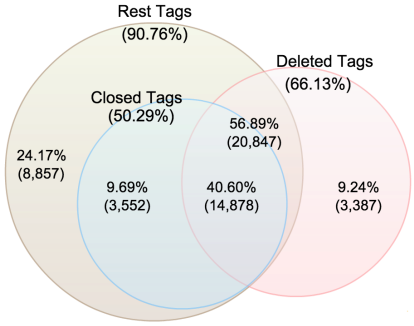


Figure 4.10: shows the venn diagram of tag distributions of questions on Stack Overflow.

questions receive at least one form of edit. Also, moderator-deleted questions receive more edits than author-deleted questions.

Figure 4.11: shows the word cloud of the top-50 tags that occur on deleted questions. 9.24% of tags are unique to deleted questions on Stack Overflow.

Table 4.6: shows edit details of deleted questions on Stack Overflow

	Moderator	**Author**	**Total**
Deleted Questions	237,163	33,330	270,493
Question Content Edited	226,286 (95.41%)	26,461 (79.39 %)	252,747 (93.44%)
Question 'Closed'	34,209 (15.12%)	2.167 (8.19%)	36,376 (14.38%)

We also notice that 14.38% of deleted questions were marked as 'closed' before they were deleted – a higher percentage of moderator-deleted questions were marked as 'closed' than author-deleted questions. Figure 4.12 shows the distribution of questions marked as 'closed' due to five reasons for both – author and moderator deleted questions. We see that a higher percentage of author-deleted questions are marked as *too localized*, *duplicate* and *off topic*. However, a higher percentage of moderator-deleted questions are marked as *subjective* and *not a real question*.

Figure 4.13 shows the distribution of percentages of questions on various edit categories for (left) 'closed' versus deleted and (right) moderator versus author deleted questions. We observe that 'closed' questions receive a higher percentage of edits than deleted questions. This signifies

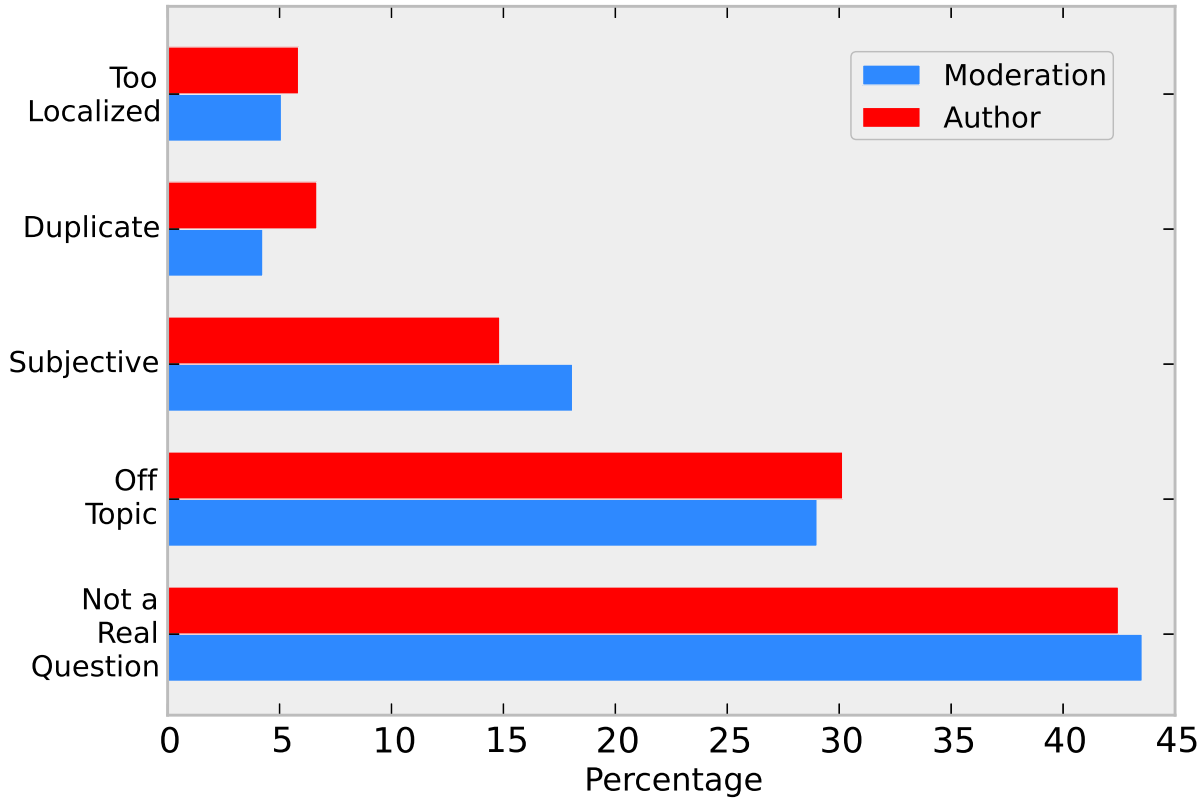


Figure 4.12: shows the distribution of questions marked as 'closed' on five reasons for author and moderator deleted questions.

that the community puts a greater effort to edit and improve 'closed' questions than it does for deleted questions. We also see that core content of a question (title and body) for author-deleted questions receive a higher percentage of edits than moderator-deleted questions. This shows that author-deleted questions are inferior in quality than moderator-deleted questions and require more work to improve their content.

Quality Pyramid

'Closed' questions are questions which are deemed unfit for the Stack Overflow format. A 'closed' question is low quality but has the potential to be improved upon. On the other hand, deleted questions are very poor in nature and beyond improvement. Prior analysis in this section revealed that deleted questions fare extremely poor on multiple community value quality indicators. We also find that some topics of deleted questions are entirely irrelevant to the Stack Overflow website. In addition, we see that the community puts in more effort to improve a 'closed' question than it does for a deleted question. These findings reveal that question quality on Stack Overflow has a pyramidal structure – regular questions lie at the top of the pyramid (good quality), followed

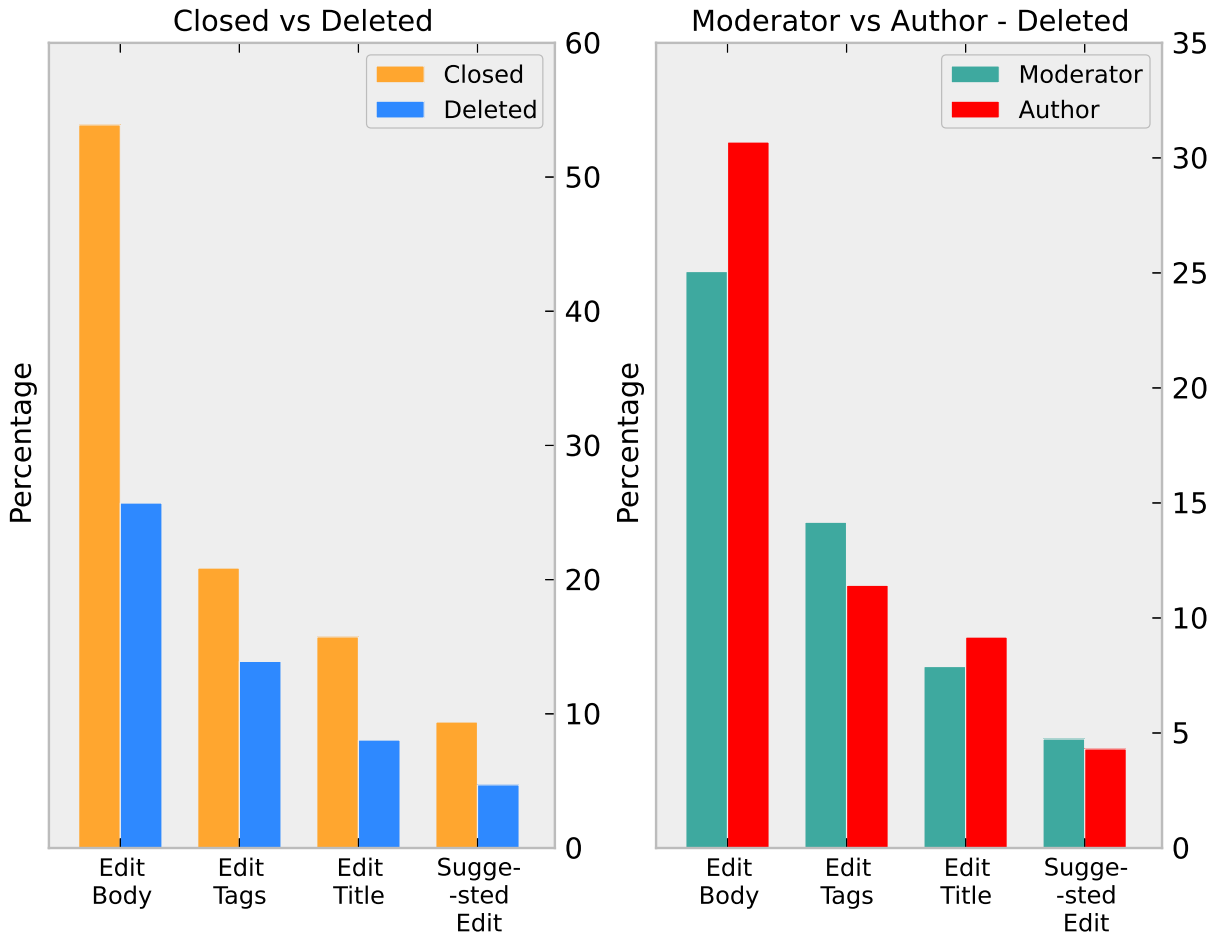


Figure 4.13: shows the distribution of edit question history on (left) 'closed' versus deleted and (right) moderator versus author deleted questions.

by 'closed' questions (bad quality) and deleted questions are placed at the bottom of the pyramid (extremely poor quality). Figure 4.14 shows this underlying question quality pyramid structure on Stack Overflow.

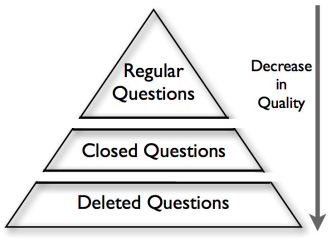


Figure 4.14: shows the underlying question quality pyramid structure on Stack Overflow.

Accidental Question Deletion

Stack Overflow provides a procedure to *undelete* a deleted question. The similar voting procedure to that of deletion is followed to *undelete* a question. Table 4.7 shows some examples of undeleted questions on Stack Overflow.

Table 4.7: shows examples of Undeleted Questions on Stack Overflow.

Id	Title	Score
145	Compressing / Decompressing Folders & Files in C#?	10
249	Accessing a remote form in php	15
5235643	Colon(:) and number in filename in Visio	1
5767118	Facebook text field detection	0

We find a total of **9,350** *undeleted* questions on Stack Overflow. **8,536** of these total questions were originally deleted by the question author while **814** questions were deleted by a moderator. We now analyze the time taken to *undelete* a question from the time of deletion. Figure 4.15 shows the percentile plot of time taken to *undelete* a question from the time of deletion of author-deleted and moderator-deleted questions. We find that most author-deleted questions are *undeleted* within 2 minutes of its deletion. We attribute this peculiar behavior to accidental deletion. The question author accidentally deletes her question but on realization of her mistake, she *undeletes* the question. We observe that moderator-deleted questions take a relatively longer time to *undelete*. This lag may be due to the time required for the *undelete* voting procedure as defined by the Stack Overflow community guidelines. The guidelines specify that a deleted question must receive a minimum of 3 *undelete* votes to be *undeleted*. This leads to an increase in time taken *undelete* moderator-deleted questions.

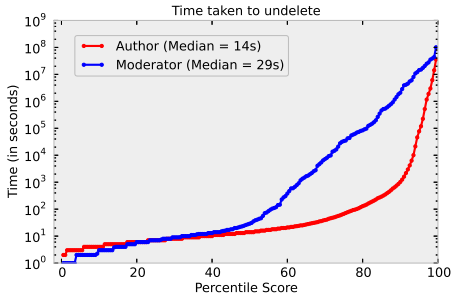


Figure 4.15: shows the percentile plot of time taken to *undelete* a question from the time of deletion of author-deleted and moderator-deleted questions.

We now analyze the *tags* on *undeleted* questions. Figure 4.16 shows the word cloud of the top-50 tags that occur in undeleted questions on Stack Overflow. We notice the presence of programming related tags like *objective-c*, *android* and *c#* which points out these *undeleted* questions are relevant to Stack Overflow.

Figure 4.16: shows the word cloud of the top-50 tags that occur in undeleted questions on Stack Overflow.

4.4 Deleted Question Prediction

In the second phase of our study, we develop a predictive model to detect a deleted question at the time of question creation on Stack Overflow. We frame the problem of deleted question prediction as a binary classification task. In our supervised classification framework, we simulate real-world conditions viz. we only use information available at question creation time. We do not have access to information on answers and other crowdsourced information like *view counts*, *favorite* votes and *question score*. In addition, Stack Overflow consists of millions of questions with thousands of topics (recall that there are 34,000+ *tags*). We also perform our experiments on the entire unadulterated dataset of Stack Overflow. All these factors make the task of prediction of a deleted question at its creation time complex and extremely challenging in nature.

Feature Identification

We experiment with **47** features based on *profile*(S_A), *community*(S_B), *question content*(S_C) and *text syntax*(S_D) for our prediction task. *Profile* based features are based on the user-generated content on the Stack Overflow website. *Community* based features are derived via the crowdsourced information generated by the Stack Overflow community. *Question content* features are based on the text and metadata of the question while *syntactic* or *text syntax* features are based on the writing

style of the text in the question. In order to investigate features based on question content, we make use of the latest Linguistic Inquiry and Word Count (LIWC2007) software [123]. LIWC2007 is a natural language psychometric analysis software that contains 4,500 words and 64 hierarchical dictionary categories. LIWC2007 takes a text document as input and outputs a score for the input over all 64 categories based on the writing style and psychometric properties of the document. We utilize the LIWC2007 software to understand the psychometric properties of natural language text in deleted questions. We find 11 LIWC categories – personal pronouns(I, them, her), pronouns(I, them, itself), space(down, in, thin), relativity (area, bend, exit, stop), inclusive (and, with, include), cognitive process(cause, know, ought), social(mate, talk, they, child), function words, conjunctions (and, but, whereas) and prepositions(to, with, above) – to be distinctive. We include these categories as features for our classification task. Table 4.8 shows all the 47 features based on four different categories. Features marked with † are generated using LIWC2007.

Note that the challenge of our supervised learning task is to predict the probability of deletion at question creation time. Hence, we only consider features which are available at the time of question creation. We understand that other features, for example – answer activity, could be a good discriminative feature for classification. But, including such features would violate the real-world conditions and therefore, we choose not to include such features in our experimental framework.

Experimental Framework

We recall that we have a total of 270,604 deleted questions as available by using the Stack Overflow database snapshots. 35,556 deleted questions do not have information about their question authors available. Since, an entire feature set in our supervised learning task is based on user profile we ignore these set of questions for this part of our study. Therefore, we have a final total of **235,048** deleted questions in the positive class of our dataset. We also see that the total number of non-deleted questions are extremely high in comparison to that of deleted questions viz. 95% of the questions are non-deleted while only 5% of these questions are deleted. Hence, the dataset is highly skewed towards the positive class. There have been various approaches in machine learning literature which deal with supervised learning for imbalanced class problems. One such popular approach is to randomly down sample the skewed or majority class data and make the dataset balanced [70]. Therefore, we randomly select **235,048** questions from the non-deleted class to form the negative class of our dataset. However, such a method may induce a sampling bias. In order to eliminate this bias, we draw 10 random samples of non-deleted questions and perform our prediction experiments on each random sample. We report our results on the average of the experiments drawn from all 10 random samples.

We experiment with multiple classifiers and find that *Adaboost* classifier gives the best performance. *Adaboost* or Adaptive boosting is an ensemble based machine learning framework which combines the performance of a series of weak classifiers to configure a strong classifier [58]. Concretely, *Adaboost* modifies subsequent classifiers in an attempt to correctly classify wrongly classified instances from previous classifiers. Prior work in content quality on community based question-answering websites have also observed best performance with ensemble-based learning

methods [27, 87]. We use a 70-30% training-testing split and perform 10-fold cross validation to prevent the problem of over fitting. We use decision tree as the base classifier and SAMME.R for the boosting algorithm [138]. The learning rate and number of estimators parameters are set to their default values. Table 4.9 shows the experimental setup details for our supervised classification task.

Evaluation

We now evaluate the performance of our classifier. Table 4.10 shows the confusion matrix for our supervised classification task. We see that our system is able to accurately classify 65.9% of deleted questions and 66.1% of non-deleted questions with an overall accuracy of 66%.

In order to understand the importance of our feature sets, we incrementally add feature sets and evaluate the performance of our classifier. Table 4.11 shows the classification performance on incremental feature sets based upon multiple evaluation metrics – F1 score, Accuracy and Area-Under-Curve(AUC). We note the improvement in performance of our classifier on each feature set. This shows that our feature sets are important to the classification task.

Feature Analysis

We now analyze our feature set in order to understand important features for deleted question prediction. The *Adaboost* classifier informs the discriminator power of each feature for classification. Figure 4.17 shows the relative importance of top 20 features for deleted question prediction as given by our classifier. We see that the top-20 features are a mixture of features from all four categories of feature sets. It points out the importance of the use of different categories of feature sets for prediction.

4.5 Conclusions

We conduct the first large scale study of deleted questions on Stack Overflow. We observe an increasing trend in the number of deleted questions on Stack Overflow over the last 2 years. The community takes significant time to detect a potential deleted question but moderators take swift appropriate action. However, we notice that moderators bear most of the encumbrance of detecting a deleted question. We also find that authors delete their own questions to salvage reputation points on the website. In general, deleted questions are extremely poor in worth to the Stack Overflow community. Also, deleted questions are significantly low in quality than 'closed' questions. We discover the existence of a pyramidal structure of question quality in which deleted questions lie at the bottom of the pyramid (extremely poor quality). We also notice accidental question deletion by authors. We build a predictive model to detect deleted questions on Stack Overflow and report 66% accurate predictions. We employ four categories of feature sets – *user profile*, *community based*, *content based* and *stylistic* features – and report most discriminatory features.

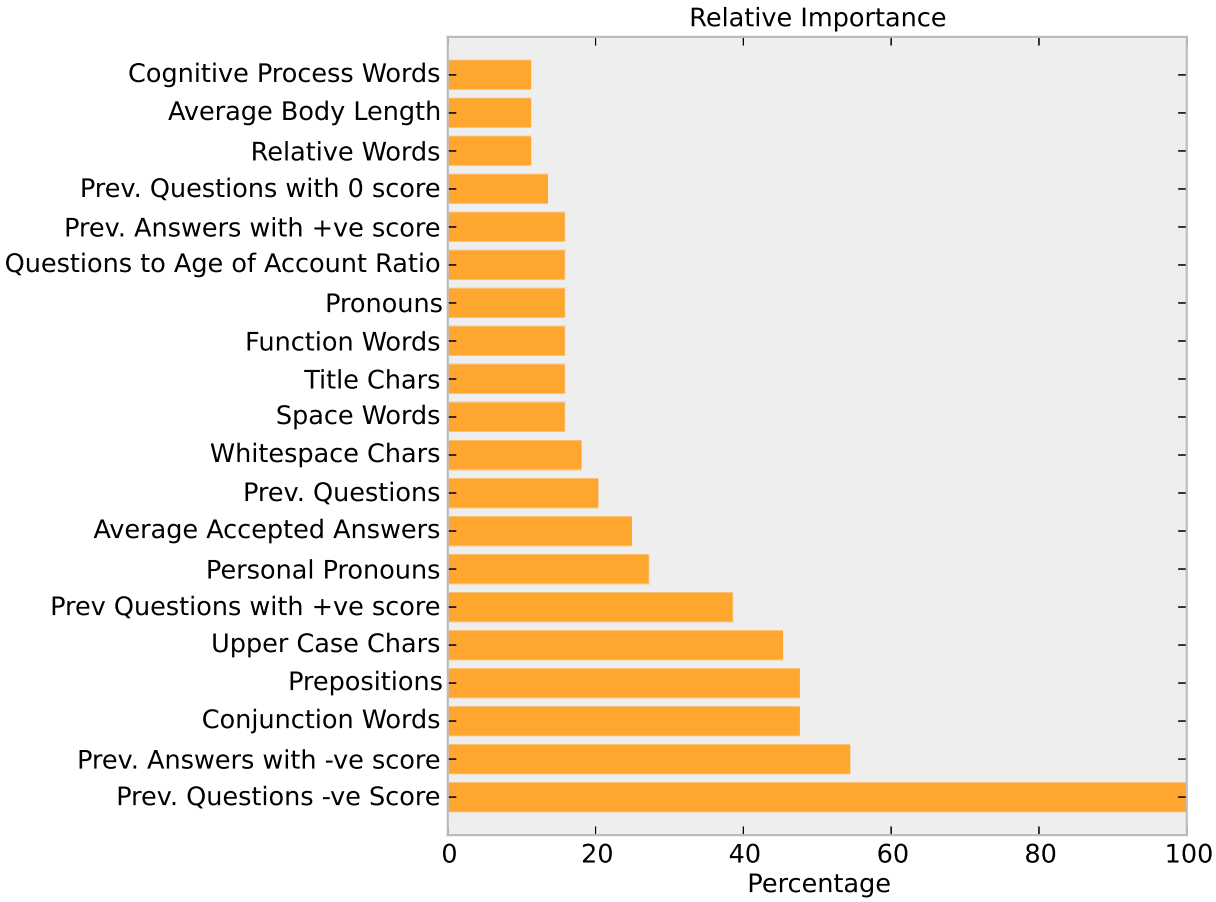


Figure 4.17: shows the relative importance of the top 20 features for deleted question prediction.

Table 4.8: lists the 47 features used for our prediction task. Each feature belongs to a specific category and features marked with † are generated using LIWC2007.

Set	Type	Features
S_A	Profile	Age of Account
		Previous Questions with -ve score
		Previous Questions with +ve score
		Previous Questions with 0 score
		Previous Answers with -ve score
		Previous Answers with +ve score
		Previous Answers with 0 score
		Number of Previous Questions
		Number of Previous Answers
		Number of Previous Badges
		Questions to Age of Account Ratio
		Answers to Age of Account Ratio
S_B	Community	Average Answer Score
		Average Question View Counts
		Average Number of Comments
		Average Favorite Votes
		Average Question Score
		Average Number of Accepted Answers
S_C	Content	Number of URLs
		Number of Previous Tags
		Code Snippet Length
		LIWC score of Personal Pronouns†
		LIWC score of Pronouns†
		LIWC score of *Space* words†
		LIWC score of *Relativity* words†
		LIWC score of *Inclusive* words†
		LIWC score of *Cognitive Process* words†
		LIWC score of *Social* words†
		LIWC score of 1st person singular pronouns†
S_D	Syntactic	LIWC score of Function Words †
		LIWC score of *Conjunctions*†
		LIWC score of Prepositions†
		Number of characters in body
		Number of alphabetical characters in body
		Number of upper case characters in body
		Number of lower case characters in body
		Number of digit characters in body
		Number of white case characters in body
		Number of special characters in body
		Number of punctuation marks in body
		Number of words in body
		Number of short words in body
		Number of unique words in body
		Average body word length

Table 4.9: shows the experimental setup for the deleted question prediction task.

Dataset	470,096 questions
Deleted (+ve class)	235,048 questions
Non-Deleted (-ve class)	235,048 questions (10 times)
Classifier	Adaboost
Learning Rate	1.0
Base Classifier	Decision Tree
Number of Estimators	100
Boosting Algorithm	SAMME.R
Cross Validation	10-fold
Classification Runs	10-times (one for each +ve/-ve pair)
Training-Testing split	70-30%
Feature Sets	$\{S_A\}$,$\{S_A, S_B\}$, $\{S_A, S_B, S_C\}$, $\{S_A, S_B, S_C, S_D\}$

Table 4.10: Confusion Matrix – Prediction Performance

		Predicted	
		Deleted	**Non-Deleted**
True	**Deleted**	**65.9%**	34.1%
	Non-Deleted	33.9%	**66.1%**

Table 4.11: shows the classification performance on incremental feature sets based upon multiple evaluation metrics – F1 score, Accuracy and Area-Under-Curve(AUC)

Feature Set	**F1**	**Accuracy**	**AUC**
$\{S_A\}$	58.91	56.81	57.19
$\{S_A, S_B\}$	61.83	59.59	61.11
$\{S_A, S_B, S_C\}$	63.38	62.17	65.24
$\{S_A, S_B, S_C, S_D\}$	65.8	66.0	70.01

Chapter 5

Integrating Issue Tracking Systems with Community Based Question-Answering Websites

In the previous two chapters, we focused on low quality content analysis and detection. We saw that there are different levels of low quality content - (1) content which can be improved (*closed* questions) and (2) content which is beyond repair (*deleted* questions). We also observed that there are multiple indicators of low quality content (Refer Figure 3.15). Two indicators of quality content are – (1) Stackoverflow URLs and (2) URLs. In the next two chapters, we focus on these two indicators to help enhance quality on software maintenance systems. Inn this chapter, we first propose a system to integrate CQA sites with Issue Tracking Systems to help software professionals during maintenance process.

5.1 Research Motivation and Aim

Research shows that software developers spend a significant amount of their time outside their development environment (such as Eclipse Integrated Development Environment) and inside their Web Browser searching and navigating information available on the Web required for problem-solving [69]. Brandt et al. investigate the role of online information and resources while programming [45]. Their case-study reveals that programmers spent 19% of their programming time on the Web to accomplish several different kinds of activities (such as learning of unfamiliar concepts, language syntax, API or function usage, reading tutorials or how-to articles, connecting high-level knowledge to implementation details) [45].

Communication, collaboration, exchanging knowledge and searching for information is not only a common activity during development but also during bug resolution and bug fixing [40][140]. Bertram et al. mention that issue tracking is a social process and issue tracking systems are a focal point for communication and coordination between stakeholders [40]. We observe that several comments posted on issue tracking system threaded discussion forums contains links to several

social-media websites (Web 2.0 platforms) such as online discussion forums and community-based question-answering websites. Web searching and navigation, referring to online resources and posting links to relevant discussions outside the issue tracking system is integral to the bug resolution process. The focus of the work presented in this paper is to study the role of community-driven question and answering websites as a knowledge resource during bug resolution. We observe links to Stackoverflow.com (a popular and widely used community-based question and answering website for programmers) in issue tracking system discussion forums for popular open source projects such as Google Chromium browser, Google Android operating system, Mozilla Firefox browser and Eclipse IDE.

Mamykina et al. conduct user interviews and perform a statistical data analysis on Stack Overflow and reveal that over 92% of Stack Overflow questions about expert topics are answered in a median time of 11 minutes [94]. Stack Overflow is popular and widely adopted not only because of a fast response time but also because of high quality answers [94]. Bacchelli et al. analyze Stack Overflow data dump (consisting of 750000 registered users, 2 million posed questions, and 4 million answers) and present a graph demonstrating a steeply increasing trend of the number of questions and answers exchanged every month on the popular Stack Overflow website Treude et al. analyze Stack Overflow data and reveal that the Q&A website is particularly effective at code reviews and conceptual questions [127]. They conduct a characterization study and uncovered different types of questions being asked on Stack Overflow such as: how-to, discrepancy, environment, error, decision help, conceptual and review [127]. Anderson et al. study the dynamics of the community activity (on Stackoverflow.com) that shapes the answers to posted questions and study how answers and voters arrive over time and how it influences the eventual outcome [31].

Zimmermann et al. propose several directions towards improvement of existing bug tracking systems and argue that current issue tracking systems are merely interface to back-end relational databases that archive reported bugs and motivate the need of extensions that can solve information need problem of developers and bug-fixers [140]. They observe that often exchange of information is stretched overtime (resulting in delays) due to the current limitations in the design and features of existing issue tracking systems[140]. The research work presented in this paper is motivated by our belief that enhancing existing bug tracking system by facilitating developers find relevant information (recommendation and information retrieval) from community-based question and answering website such as Stackoverflow.com can reduce the context-switch (due to plug-ins within the issue tracking system) and time spent by developers in manually searching and filtering relevant content. We believe that integration of issue tracking system with social-media Web 2.0 platforms specifically for programmers is a relatively unexplored area and our motivation is to throw light on this topic by conducting an in-depth empirical study (qualitative and quantitative) on real-world representative dataset.

The specific research aim of the work presented in this paper is the following:

1. To investigate the role and extent of adoption of social-media based and community-driven Q&A websites (such as Stack Overflow, Super User and Server Fault) in a knowledge-intensive and collaborative activity of software bug resolution. The aim of the study presented in this paper is to conduct an exploratory study on the topic of integrating issue track-

ing systems with community-based Q&A websites.

2. To examine the correlation and impact of the presence of links to community-based Q&A website in issue tracking system threaded discussion forum on the mean time to resolve a bug and number of comments.

3. To investigate solutions for recommending Stack Overflow Q&A for an incoming bug report based on textual and contextual (metadata) features.

4. To conduct a survey and discussion with practitioners (bug fixers, bug reporters and QA managers) on the need, challenges and directions for integrating issue tracing system with community driven Q&A websites.

Research Contributions

In context to existing work, the study presented in this paper makes the following novel contributions:

1. The work described in this paper is the first focused study on integration of issue tracking systems with community driven question and answering websites such as Stackoverflow. While there has been work in the area of code-editors & development environment integration (Section **??**) with Stackoverflow as well as code-editor integration with web-search and external websites (Section **??**), the integration of Stackoverflow with issue tracking systems is a unique research direction.

2. We present experimental results (based on a series of experiments conducted on publicly available dataset from two popular, large, complex and open-source projects: Google Chromium and Android) indicating presence of several links to Stackoverflow question & answers facilitating the process of bug resolution. We conduct a characterization study and present our perspective on the correlation between Stackoverflow references and mean time to repair a bug, top domains in issue tracking system threaded discussion forums and illustrative examples showing links to various Web 2.0 platforms in addition to Stackoverflow.

3. We present a solution based on analyzing textual features (textual similarity between bug report title and Stackoverflow questions) and contextual features (such as question tags representing the topic) to recommend a Stackoverflow question in response to a bug report. We believe that a recommendation engine (tool support) that automatically suggests relevant Stackoverflow knowledge-base to developers can save time during bug resolution. We present the proposed solution and empirical results ((performance evaluation and validation)) demonstrating the effectiveness of the method on dataset containing the ground-truth.

4. A survey conducted with experienced Software Maintenance Professionals on the topic of integrating issue tracking system with community driven Q&A websites. We believe that

there is a dearth of academic studies surveying the needs, problems encountered, human-factors and suggestions on the problem area discussed in this paper. To the best of our knowledge, the work presented in this paper is the first study to conduct a survey of bug reporters and fixers on the topic and perform empirical analysis on a real-world publicly available dataset from a popular open-source project.

5.2 Survey of Software Maintenance Professionals

The research work presented in this paper is motivated by the need to investigate solutions to common problems encountered by Software Maintenance Professionals. We believe that inputs from practitioners are needed to inform our research. Table 5.1 displays the results of a survey (note: while the paper is under review - we are receiving responses and will be sending survey to more software maintenance professionals) of Google Chromium Developers. We sent an email to the Google Chromium Developer mailing list (chromium-dev@chromium.org) and received 13 responses. We also received few replies to our email as well as free-form text response in addition to the response to multiple-choice questions in Table 5.1. As shown in Table 5.1, 23.1% of the survey respondents have $1-5$ years of work experience in Software Maintenance whereas a significant percentage (76.9%) had more than 5 years of experience in the field. Our research shows that as a bug-fixer, 61.5% search and post questions (moderate and high usage) on community-driven Q&A websites like StackOverflow to leverage the archived knowledge and get inputs from experts. 38.5% believe that linking community-driven Q&A websites like StackOverflow with the threaded discussion of issue tracking system can decrease the mean repair time or decreases the bug resolution time. We notice that the usage of StackOverflow is more by bug fixers than bug reporters.

One of the replies we received (presented verbatim but removed the name and email of the person) is: "Chrome extension developers are encouraged to consult Stack Overflow [1], as it's now an official support channel for Chrome extension developers (and will be THE future official support channel)". Another response we received is: "Sites like StackOverflow tend to be used for more general issues. I do consider them a very important tool related to development, and might find them useful if I worked on development tools, or a general runtime environment. I've never observed a site like StackOverflow being used to address issues specific to a project I was working on". Frequently Asked Questions [2] on Google Chrome Extensions mentions "If you don't find an answer to your question here, try the Chrome Web Store FAQ, the [google-chrome-extension] tag on Stack Overflow, the group, or the store help". Facebook and Stack Exchange are partners to support the Facebook developer community and facebook.stackoverflow.com is an official developer support channel for Facebook developers [3]. Andorid Developers blog[4] mention that that they

[1] https://developer.chrome.com/extensions/faq.html
[2] https://developer.chrome.com/extensions/faq.html
[3] http://blog.stackoverflow.com/2011/08/facebook-stackoverflow/
[4] http://android-developers.blogspot.in/2009/12/hello-stack-overflow.html

Table 5.1: Survey Results: Google Chromium Developers on Integration of Issue Tracking System with Community-Driven Q&A Websites

Q1: How many years of work experience you have in the area of Software Maintenance?	
Less than 1 year	0.0%
Between 1 and 5 years	23.1%
More than 5 years	76.9%
Q2: As a software maintenance professional (bug reporter, triager, bug fixer, QA manager, developer) do you observe Community-Powered Q&A Websites for programmers such as StackOverflow (http://stackoverflow.com/) being used by bug reporters and bug fixers to search and post questions?	
I observe a negligible or low usage	61.5%
I observe a moderate usage	38.5%
I observe an increasing trend and significant adoption [high usage]	0.0%
Q3: As a bug-reporter - do you search community-driven Q&A websites like StackOverflow (http://stackoverflow.com/) before reporting a bug and link Q&A artifacts as part of the bug report [when applicable]	
No (or low usage)	61.5%
Sometimes (moderate usage)	30.8%
Always (or high usage)	7.7%
Q4:As a bug-fixer - do you search and post questions on community-driven Q&A websites like StackOverflow to leverage the archived knowledge and get inputs from experts [when applicable]	
No (or low usage)	38.5%
Sometimes (moderate usage)	53.8%
Always (or high usage)	7.7%
Q5:Do you think link(s) to community-driven Q&A websites like StackOverflow in the threaded discussion of issue tracking system decreases the mean repair time or decreases the bug resolution time	
No significant or noticeable impact	61.5%
Some impact (as relevant and useful information is integrated - saves developers time)	38.5%

Table 5.2: Questions, Answers, Answered and Users Statistics of StackExchange Technology Sites on July 1st 2012. SO: Stack Overflow, SF: Server Fault, SU: Super User, AE: Android Enthusiasts, PR: Programmers, UE: User Experience, AU: Ask Ubuntu, WM: Webmasters, AP: Ask Different (Apple Products)

	SO	SF	SU	AE	PR	UE	AU	WM	AP
Questions	3.3m	118k	127k	8.1k	19k	4.7k	55k	9.8k	17k
Answers	6.6.m	240k	235k	12k	94k	16k	81k	18k	31k
Answered	80%	83%	80%	84%	99%	100%	79%	93%	84%
Users	1.2m	86k	108k	14k	51k	14k	61k	14k	21k

believe Stack Overflow can improve developer support and particularly useful for developers new to Android. Android tag on Stack Overflow is an official Android app development Q&A medium and the blog advices to the Android developer community that they are encouraged to post their beginner-level technical questions there. Google TV project[5] mentions Stackoverflow.com as a developer resource encouraging developers toask questions and share expertise and wisdoms. JQuery developer support page[6] mentions "Developers can likely find an answer (on Stackoverflow.com) for whatever issue they are experiencing and if their question is not addressed then they can ask a new question and often receive a quick response".

5.3 Empirical Analysis

Experimental Dataset

We conduct experiments on publicly available dataset so that our results can be replicated and used for benchmarking by other researchers. We chose Google Chromium and Android issue tracking dataset as both the projects are open-source, large and complex systems. Both are long-lived, popular and widely used browser and mobile operating system respectively. We conduct exactly same analysis on two different projects to remove bias and increase generalization of conclusions. Stackoverflow dataset is publicly available and both Google issue tracker and Stackoverflow dataset can be accessed through programming APIs. Table 5.2 displays the number of questions and answer statistics on Stackexchange.com domain (Stackoverflow is one of the Stackexchange site) during the time-frame of our study. Table 5.3 displays the experimental dataset details for the Google Chromium and Android projects. As shown in Table 5.3, the number of downloaded and closed issues for Android is 36262 and 13624 respectively and the number of downloaded and closed issues for Chromium is 142175 and 104814 respectively. The dataset spans across multiple years: 2007 to 2012 for Android and 2008 to 2012 for Chromium.

[5] `https://developers.google.com/tv/web/`

[6] `http://learn.jquery.com/about-jquery/additional-support/`

Table 5.3: Experimental dataset details (open-source Android and Chrome projects)

	Android	**Chrome**
Download Date	August 14, 2012	August 12, 2012
Num. Issues Available on Issue Tracker	36286	142175
Number of Issues Downloaded	36262	134410
Issues deleted or restricted access	45	7765
Number of Open Issues	22638	29596
Number of Closed Issues	13624	104814
Num. Issues Label = Declined	6615	0
Num. Issues Label = Duplicate	1422	28460
Num. Issues Label = Spam	1180	0
Number of Distinct Owners	134	933
Timestamp of first bug report	2007-11-12 16:02:55	2008-08-30 16:00:21
Timestamp of last bug report	2012-08-13 08:56:16	2012-08-11 21:16:59

Table 5.4: Number of bug reports in experimental dataset containing links to relevant Stack Exchange sites

	Android		**Chrome**	
SE Domain	**Desc.**	**Desc+Comm.**	**Desc.**	**Desc+Comm.**
SO: Stack Overflow	216	448	269	418
SF: Server Fault	0	0	0	3
SU: Super User	0	2	10	36
AE: Android Enthusiasts	9	16	0	7
PR: Programmers	0	0	0	0
UE: User Experience	0	0	0	1
AU: Ask Ubuntu	1	1	7	10
AP: Apple Products	0	0	2	2

Characterization of Issue Reports and Stackoverflow links

We compute the number of bug reports in the experimental dataset containing links to 8 Stackexchange websites. Stackexchange consists of 89 question and answer website (at the time of writing this paper). Table 5.4 shows the Stackexchange domain, number of bug reports containing links to the Stackexchange domain in the issue description and number of links to the Stackexchange domain in the issue description or comments for both the Android and Chromium dataset. Table 5.4 reveals that Stackoverflow is the most referred in contrast to other relevant Stackexchange sites. Table 5.4 indicates that there are several issue reports (in both the projects) containing links to community-driven question and answering website supporting as an evidence of the role of

Table 5.5: Top domains in issue tracking system based on the number of links in DESC: Description and D+C: Description and Comments

Android				Chrome			
Domain	DESC	Domain	D+C	Domain	DESC	Domain	D+C
developer.android	1338	code.google	3499	build.chromium	5788	src.chromium	479007
code.google	581	developer.android	1926	code.google	3065	codereview.chromium	70643
groups.google	340	google	1419	google	2609	chromiumcodereview	17597
schemas.android	307	groups.google	654	codereview.chromium	1400	code.google	13455
google	223	forum.xda-developers	583	chrome.google	1161	build.chromium	12726
stackoverflow	216	review.source.android	459	src.chromium	1100	bugs.webkit	8095
dl-ssl.google	147	stackoverflow	448	trac.webkit	972	trac.webkit	6693
android.git.kernel	105	schemas.android	407	youtube	935	google	5745
youtube	94	kernel.android.git	383	jsfiddle	858	crbug	5009
source.android	89	dl.ssl.google	310	W3	837	dev.chromium	2703
forum.xda-developers	84	market.android	306	bugs.webkit	795	chrome.google	2541
github	75	youtube	245	dev.chromium	762	goto.google	1800
en.wikipedia	70	android-review	239	test-results.appspot	467	chromium	1667
market.android	50	github	236	en.wikipedia	398	youtube	1563

such websites in bug resolution process. While the focus of the work presented in this paper is on integration of issue tracking system with community-based question and answering websites for programmers, we uncover the top URL domains referred in the threaded discussion of Google Android and Chromium issue trackers. Table 5.5 shows the Top 14 URL domains and the number of bug reports mentioning it in the description and both description and comments. Table 5.5 indicates that several Web 2.0 platforms are referred in issue tracker discussion forums and activities like web-search and navigation is integral to communication involving bug resolution.

We notice that several links to micro-blogging websites such as Twitter[7] and video-sharing website such as YouTube[8] are present in issue tracking system discussion forums. While the focus of the work presented in this paper is the role of community-driven question and answering website during bug resolution, we found evidences indicating that various other Web 2.0 platforms and popular social media websites are also frequently referred during developer discussion during bug fixing. Table 5.6 shows illustrative examples of bug reports and developer comments containing references to Twitter and YouTube. We observe that (as evident from the developer comments in Table 5.6) YouTube is commonly used to create a video of user interaction with the system to demonstrate steps to reproduce and actual system behavior.

One of the research question or aim of our study is to investigate correlation between presence

[7] www.twitter.com
[8] www.youtube.com

Table 5.6: Developer comments referring to tweets on Twitter and videos on YouTube

	Dataset	Social Media	Issue ID	User Comments
1	Android	Twitter	13674	Samsung have confirmed an OTA update in the next few days: [Twitter-URL]. Is reboot issue also fixed in this update?
2	Android	Twitter	6914	I managed to get a reply from RG on Twitter. Here's what he stated [Twitter-URL]
3	Android	Twitter	3389	I've seen some reports that Android 2.3.4 for the Google Nexus now has settings that indicate support for IPv6 on the radio interface [Twitter-URL]
4	Android	YouTube	22956	This is a reproducible issue. I've made a video to demonstrate the problem: [YouTube-URL]
5	Android	YouTube	30035	I made a video here to demonstrate the bug better: [YouTube-URL]
6	Android	YouTube	28016	I have been trying to fix this issue for days now, trying many different things, and today I came across a video that explains why wrap_content is so bad. Here's the link: [YouTube-URL]
7	Chromium	Twitter	125304	It seems that Typekit has recently added support for Chrome OS: [Twitter-URL]
8	Chromium	Twitter	112367	A few years ago the Adobe Flash community got together and pressured Adobe to add mic recording capability to Flash with the getmicrophone project: [Twitter-URL]
9	Chromium	Twitter	107949	I have also confirmed someone else is experiencing the same problem [Twitter-URL]
10	Chromium	YouTube	132090	I made this youtube video that demonstrates this bug's behavior and crash: [YouTube-URL]
11	Chromium	YouTube	132032	Here is a link to a video of this behavior: [YouTube-URL]
12	Chromium	YouTube	127598	I uploaded a recording of how to reproduce (the bug), though [YouTube-URL]

Table 5.7: Mean time to repair bug with and without StackOverflow (SO) link in the bug report description

Project	Dataset	Mean	Std. Dev.	Median
Android	All	81 Days, 23:25:21	165 Days, 14:27:54	9 Days, 22:03:96
	With SO URL in Desc.	39 Days, 00:56:19	79 Days, 13:26:07	6 Days, 20:20:10
	Without SO URL in Desc.	83 Days, 13:35:30	67 Days, 9:04:56	10 Days, 4:34:16
Chrome	All	219 Days, 22:14:21	303 Days, 19:17:57	42 Days, 23:33:56
	With SO URL in Desc.	350 Days, 10:05:56	278 Days, 20:48:07	411 Days, 10:31:50
	Without SO URL in Desc.	219 Days, 17:42:20	303 Days, 20:02:31	42 Days, 22:50:48

Table 5.8: Number of comments in a bug report with and without StackOverflow link in the bug report description

Project	Dataset	Mean	Std. Dev.	Median
Android	All	5.58	55.83	1.0
	With StackOverflow URL in Desc.	3.2	6.3	1.0
	Without StackOverflow URL in Desc.	5.6	55.97	1.0
Chrome	All	8.37	11.09	6.0
	With StackOverflow URL in Desc.	5.85	5.84	4.0
	Without StackOverflow URL in Desc.	8.37	11.09	6.0

of Stackoverflow links (as a representative example for the class of community-driven question and answering websites) in issue tracker discussion forum and the mean repair time for bug reports and the number of comments posted till closure of a bug report. Table 5.7 indicates that the resolution time (time difference between bug report and closure) of bug reports (excluding cases such as spam, duplicate, declined, unreproducible and invalid from the datatset) for issues containing links to Stackoverflow is less in contrast to reports not containing Stackoverflow links for the Android project. The same phenomenon is observed with respect to the number of comments (refer to Table 5.8) for both Android and Chromium projects. Table 5.7 and 5.8 shows the same trend (number of comments in the threaded discussion) for both the projects (Table 5.7 and 5.8 displays the mean, standard deviation and median). One explanation is that because a Stackoverflow Q&A is itself like a threaded discussion where the developer community can answer questions (there can be multiple answers and perspective to a question), rate answers based on the quality and reply to each other, the available knowledge-base may result in a quicker bug resolution and decrease the number of comments.

We perform a manual inspection of the issue reports which contains a link to Stackoverflow website to examine the extent of textual similarity between the titles of the bug report and the Stackoverflow question. Our intuition is that textual similarity (term overlap) between issue reports and sackoverflow Q&A can be exploited as linguistic features for developing a Stackoverflow link and recommendation engine (automating the task of manually searching and navigating relevant Stackoverflow artifacts). Table 5.9 shows examples of few bug reports from Android issue tracking system and the title of the linked Stackexchange question (examples in which there is a link to Stackexchange Q&A in issue tracker discussion forum). Table 5.10 shows examples of a different type than Table 5.9: cases in which there is a link to an issue report in Stackexchange comment which is different than having a link from issue report to Stackexchange. Tables 5.9 and 5.10 reveal that there are some terms which are common between the title of the issue report and Stackexchange question (the common terms between the two texts which are getting compared are highlighted in bold).

Table 5.9: Textual Similarity (term overlap marked in bold) between titles (of issue report and SE questions) for cases in which there is a link from a issue tracker comment to SE Q&A

	Issue ID	Issue Report Title	SE Q&A ID	SE Question Title
1	22665	ICS: SDK missing windows **MTP driver**	3246	"**MTP** USB Device" **driver** error (screenshot) when connecting my Galaxy S to my PC in Kies mode - How can I resolve this problem?
2	23300	**Galaxy Nexus** 4.0.2, Verizon - Erratic **vibration** for both haptic feedback and **notifications**	17135	**Notification Vibration** Bug on **Galaxy Nexus**?
3	30198	Add support for **USB** 'device mode' to **android**.hardware.**usb**	22810	**USB** host port via **Android** accessory protocol adapter
4	34833	Android 4.1 **DatePickerDialog** Mulfunction	11444238	Jelly Bean **DatePickerDialog** - is there a way to cancel?
5	24746	ICS browser loses some **input** CSS **styling** on **focus**	9005550	**Input** elements on android 4.x can not be **styled** when **focused**

Table 5.10: Textual Similarity (term overlap marked in bold) between titles (of issue report and SE questions) for cases in which there is a link from a comment in SE Q&A to an issue report

	Issue ID	Issue Report Title	SE Q&A ID	SE Question Title
1	23262	no **Wifi** on **Nexus S** after upgrading to **ICS** (4.0.3)	17276	**WiFi** on my **Nexus S** no longer works after update to **ICS** - How can I troubleshoot or fix this?
2	3678	**Browser** does **not** encode urls	14370	Webpage **not** loading in the Android **browser**
3	7048	Outgoing **Call Waiting Notification**	14333	How do I enable **call waiting notification** when I am calling another person?
4	20201	since the latest update on my incrdible, my phone has displayed the **"low on Space" warning**	15666	**"Low on space" warning** with over 50% free
5	6458	Enhancement: **Undo**	5948	Is there any **undo** command when editing any field

Table 5.11: Motivating examples from Chromium browser dataset showing lexical similarity between the query and relevant result which is ranked lower in the search result

Motivating Example 1	
BR Title	Cannot open any HTML select element drop-down with the keyboard
Relevant SO QA	How can you programmatically tell an HTML SELECT to drop down (for example, due to mouseover)?
Rank	81
Top Result	keyboard navigation for ajax drop down
Motivating Example 2	
BR Title	Python test server output must be unbuffered so it doesn't corrput gtest output
Relevant SO QA	Python output buffering
Rank	117
Top Result	Unbuffered subprocess output (last line missing)

Stackoverflow Q&A Link Prediction and Recommendation

Our goal is to investigate solutions to automatically recommend relevant StackOverflow Q&A for a new bug report as we believe that such a recommendation engine can save developer time in manually searching (formulating a search string and changing context to StackOverflow website for querying the search string) and browsing the search result to extract the relevant Q&A artifact. The recommendation engine can post relevant Stackoverflow Q&A as a comment on the issue tracker discussion forum. We conjecture that the number of references to community-driven Q&A website, Web 2.0 platforms and social media websites (refer to Tables 5.4, 5.5, 5.7 and 5.8) are evidences of the value of such sites as knowledge-bases during bug resolution process and the barrier to adoption to such sites can be lowered by making it more convenient for developers to leverage the content (for example by automating the process for information retrieval). The task of recommending relevant Q&As can be formulated as an information retrieval problem in which the input to the search engine is a query containing keywords from the bug report title or description and the document-set to be searched represents the archive of Q&As indexed by the website. However, the stated problem is non-trivial and poses several technical challenges.

We demonstrate the technical challenge with the help of two motivating examples shown in Table 5.11. We select two recent bug reports containing links to Q&A on Stackoverflow and manually search the Stackoverflow.com website by giving the title of the bug report as the query string. As shown in Table 5.11, we observe that the relevant result (the Stackoverflow Q&A posted as a comment in response to the bug report) is ranked as 81 and 117 respectively for the two illustrative examples in our study. We observe that rank of the relevant result is not in Top 10 or Top 25 despite some term overlap between the title of the bug report (query string) and the title of

Table 5.12: Accuracy (precision and recall at a predefined cut-off or rank) results for the search with and without tag-based contextual features (result demonstrating baseline results and improvements over baseline).

	Android	Android [Tag]	Chrome	Chrome [tag]
Total Instances	448	448	418	418
Inst. available for Download	404	404	230	230
Does not appear in results	137/404 = 33.8%	132/404 = 32.67%	99/230 = 43.04%	119/230 = 51.73%
Does appear in results	267/404 = 66.2%	273/404 = 67.57%	131/230 = 56.95%	111/230 = 48.26%
Top 10	134/404 = 33.16%	143/404 = 35.39%	68/230 = 29.56%	57/230 = 24.78%
Top 20	149/404 = 36.88%	163/404 = 40.34%	75/230 = 32.60%	59/230 = 25.65%
Top 100	191/404 = 47.27%	217/404 = 53.71%	99/230 = 43.04%	76/230 = 33.04%
Mean (of Top 5000)	268 Rank	212 Rank	340 Rank	203 Rank
90th Percentile (of Top 5000)	1040 Rank	615 Rank	1031 Rank	421 Rank

the relevant Stackoverflow Q&A.

We compute the textual overlap between the title of the bug report and the title of the Stackoverflow question for the ground-truth dataset (issue reports containing references to Stackoverflow Q&A). We measure textual overlap in-terms of the percentage of terms in the title (after removal of non content bearing stop-words) present in the title of the referred Stackoverflow question. Similarly, we also measure the textual overlap between the title of the bug report and the tags of the linked Stackoverflow question. We observe that the textual overlap (lexical similarity) between the titles is 15.16% and 14.39% for the Android and Chromium dataset respectively. The textual similarity between the bug report title and associated Stackoverflow question is 6.2% and 5.37% for the Android and Chromium dataset respectively. The results indicate that while there is some textual overlap ($5\% - 15\%$) between the bug report title and relevant Stackoverflow Q&A title and tags, the overlap is less than 20% motivating the need to investigate non-lexical features (contextual features) and semantic search techniques for the recommendation engine.

We conduct an experiment in which we provide the title of the issue reports (in the ground-truth dataset) as a search string to the Stackoverflow Search API[9] and measure the rank of the relevant Q&A (the one referred in the issue report discussion forum) in the search result. Automatically searching the Stackoverflow.com website with just the title of the issue report and presenting Top K result to the bug reporter or collaborators serves as a naive (or baseline) recommendation engine. Table 5.12 displays the result of our experiment and shows the precision and recall (standard performance metrics for Information Retrieval systems) values for the baseline as well as improved system (application of tag-based contextual features). We can retrieve only Top 5000 search results for a given query due to the limitations of the Stackoverflow API and hence we can measure the exact rank of a given Q&A ID only if it appears in Top 5000. The total instance column (refer to Table 5.12) shows the unique number of issue report IDs and the linked Stackoverflow Q&A. We refer to the number of records as instances since one bug report can have link to more than one Stackoverflow Q&A (all are relevant and each unique pair represents an instance).

As shows in Table 5.12, we show the number of cases on which we can perform experiments

[9] `http://api.stackoverflow.com`

as not all Stackoverflow Q&A were downloadable (either the referred Q&A is removed or unable to fetch through the API). We observe that for 33.6% and 57% of the instances (Android and Chromium dataset respectively), the relevant search result was not in Top 5000 (refer to Table 5.12) when the title of the bug report was provided as the query string to the Stackoverflow Search API. Stackoverflow contains millions of Q&As and we notice that thousands of question titles have some lexical similarity with the title of the issue report in the ground-truth dataset and hence just searching the Stackoveflow archive with the issue report title as query string is not enough to retrieve relevant results. For the Android project dataset, we observe that for 66.4% of the issue report in the ground-truth dataset, the relevant result was ranked in Top 5000. However, the precision at Top 10, 20 and 100 (Android dataset) is 33.16%, 36.88% and 47.27% respectively. Table 5.12 displays the mean and 90th percentile rank (of results in Top 5000) for the baseline as well as for the system which is an improvement over the baseline.

Stackoverflow.com allows askers to tag[10] (a key-word or label) their questions which helps in question categorization, search (easier to find information which is labeled with meaningful tags) and makes it easier for subject-matter experts to listen to tags of their interest. We hypothesize that tags can be exploited as contextual features (as tags are meta-data) to improve the rank of the relevant document in the search result. We conduct an experiment to investigate if certain tags are more frequent than others (for issue reports in the ground-truth dataset and the associated Stackoverflow.com Q&A) so that tag information can be used to increase or decrease the relevance score of a Q&A with respect to a given issue report title. We notice that 392 out of 404 Stackoverflow question contains the tag "Android" and 41, 32 and 23 contains the tag Java, Eclipse and Webview respectively. Experimental results reveal that there are certain tags which have frequency count of less than 20: Android-emulator, ADT, Ice-cream-sandwich, Android-fragments and Listview have a frequency of 13, 11, 10, 10 and 9 respectively (for Android dataset). Similar tag distribution trend is observed in Chromium dataset. We observe that out of 230 questions in the ground-truth dataset: the frequency of Google-Chrome, Javascript, CSS, HTML, Google-Chrome-Extension, HTML5, JQuery, CSS3, Webkit is 109, 72, 31, 28, 22, 17, 13, 10 and 9 respectively.

We hypothesize that tag (meta-data and contextual feature) can be used as additional information (in addition to issue report title) while searching for relevant Q&A. We conduct another set of experiments in which we programmatically search the Stackoverflow.com website using issue report title as a query string and also specifically mention the most frequently occurring tag as a parameter (advance search option) to investigate improvements (from the baseline) in the precision and recall of the information retrieval (recommendation engine). Table 5.12 displays the experimental results for the baseline (only title as query string) and the enhanced system (using tag information as contextual features). We observe a minor gain in relevant results retrieved for Android (267 to 273) while a decrease in gain (131 to 111) for that that of Chromium. The decrease in gain of relevant retrieved results for Chromium is likely due to the nature of the Chromium project which encompasses aspects which are not unique to chromium like Javascript, CSS3 or HTML5 leading to linkage of more generic questions in the issue tracking system to Stackoverflow questions. However, we observe a reduction in the ranks of the documents retrieved both for

[10] `http://meta.stackoverflow.com/tags`

Android and Chromium. The mean rank for Android decreased from 268 to 212 while the 90th percentile rank decreased from to 1040 to 615. The mean rank for Chromium decreased from 340 to 203 while the 90th percentile rank decreased from to 1031 to 421. In order to confirm, if the query with title and tag yields better ranks than just title (baseline), we perform a statistical significance test. Since we do not exactly know the underlying distribution of our ranks, we choose Wilcoxon Signed Rank Test as our statistical significance test. Wilcoxon Signed Rank Tests is a non-parametric paired difference hypothesis test which explains if the population mean-ranks differ. The Null Hypothesis for Wilcoxon Signed Rank Test says that there is no difference between the two systems. The p-value for the Android dataset is $1.00530788183 * 10^{-08}$ and for the Chromium dataset is $3.86976175202 * 10^{-05}$. For both datasets, $p < 0.01$ and hence we reject the H0 with 99% confidence i.e. rank differences are significant viz. title and tag is better than just title.

5.4 Limitations and Conclusion

The survey results (Table 5.1) has 13 respondents (Software Maintenance Professionals working on Google Chromium open-source project) and can be influenced by bias (based on individual experiences and beliefs). In future, we plan to conduct survey of more professionals (across multiple projects) which will result in reducing bias and generalizing our conclusions. The experiments are conducted on a dataset belonging to two projects (Google Chromium open-source browser and Android operating system) collected for specific duration. More experiments can be conducted on dataset belonging to diverse projects (larger dataset, multiple projects of different types) to investigate the generalizability of the conclusions.

Community-based Q&A websites such as Stackoverflow.com contains answers to millions of questions (serving as an archive of knowledge-base) on various topics relevant to programmers and is widely used by software developer community. The study presented in this paper shows evidences of references to Stackoverflow Q&As being posted as comments in threaded discussions of issue tracking systems of popular open-source projects such as Google Android and Chromium. Our study reveals that issue tracking system contains links (posted by bug reporters and project team responsible for bug fixing) to several Web 2.0 platforms and social media websites (such online discussion forms, video sharing websites and micro-blogging websites). A quantitative analysis and manual inspection of developer comments around referenced URLs shows the usefulness of such websites during bug resolution. Experimental results indicate that the number of comments in response to bug reports containing references to Stackoverflow is less in contrast to number of comments in response to bug reports which does not contain links to Stackoverflow. We conduct a series to of experiments to measure the textual similarity between issue report title and the associated Stackoverflow question title and tags motivated by the need to develop a recommendation engine to automatically suggest relevant Stacoverflow.com Q&A for a given bug report. We present results showing precision and recall values for a baseline or naive recommendation engine which consists of using only the title of the issue report as a query string and demonstrate an improved accuracy result by using tags as contextual features.

Chapter 6

Samekana: A Browser Extension for Including Relevant Web Links in Issue Tracking Systems

In the last chapter, we saw how CQA sites can be integrated with Issue Tracking Systems to improve software maintenance productivity. We also observe that software professionals extensively browse the Internet to search for relevant content. In this chapter, we aim to bridge the gap and reduce the context switch during such Internet activity. We recall that URLs are a good indicator of content quality (Refer Figure 3.15). We deploy this system online system to manage and share URLs during their daily work flow. Concretely, we analyze URL sharing patterns in Issue Tracking Systems and build a web browser plugin to help software maintenance professionals organize their web references.

6.1 Research Motivation and Aim

Issue Tracking Systems (commonly referred to as Bug or Defect Tracking Systems) such as Bugzilla, Google Issue Tracker, JIRA and Mantis provides a platform to support software maintenance activities such as issue reporting, tracking and resolution. Issue (such as a bug report or a feature enhancement request) tracking and resolution is a social and collaborative process in which an issue tracker serves as a communication hub and channel between the users, developers and QA (Quality Assurance) team [40]. Previous research consisting of surveys and feedback from developers indicate the need of enhancing the capabilities and usability of existing issue tracking systems by providing additional tool support and functionality solving problems encountered by the users [77, 140]. The study by Just et al. recommend the need for providing tool support to collect and prepare information that developers need and making it easy or encouraging users or collaborators to submit details to bug reports [77]. Issue resolution is a collaborative and knowledge intensive activity. Modern Issue Tracking Systems contain a threaded discussion forum or a message board for collaborators to post comments and share knowledge.

Web searching, browsing and navigation, referring to online resources and posting web-links to resources is not only done during programming but also during issue resolution process. Previous research shows that developers spent considerable amount of time outside their development environment and in their web-browser resulting in a context-switch between the development environment and browser [64] [57] [54] [35] [51] [44]. We observe that issue reports as well as several comments posted on issue tracking system threaded discussion forums contains web-links (serving as citations and pointers to knowledge-sources) to several websites. Posting of web-links and URLs to issue reports as well as issue tracker threaded discussion forum is a common activity by developers. The broad research motivation of the work presented in this paper is to extend the capabilities of an existing web-based issue tracking system by providing tool support (through a browser extension) for issue tracker users (bug reporters, bug fixers and collaborators) enabling them to conveniently collect and organize reference to knowledge sources (refereed to during bug resolution) and include it in their comment post.

1. To study how frequently bug reporters and bug fixers conduct web-search before reporting an issue or resolving a given issue. To conduct surveys of software maintenance professionals across several projects for the purpose of investigating the usefulness of web-links on issue tracking discussion forums and their perceptions on the value of a tool to facilitate web-link saving, organizing and posting.

2. To conduct an in-depth and focused characterization study on web-links in issue tracking discussion forum. To investigate the extent of presence of web-links in issue reports and issue tracker discussion forum, top domains referred in the issue tracker, types and categories of web references and perform statistical analysis such as correlation with bug-fix time and distribution across different issue types (such as security, crash, clean-up and regression)

3. To develop a tool support (as a browser extension) for bug reports, bug fixers, bug triagers and quality assurance manager facilitating in easy saving, organizing and posting of web-links.

Research Contributions

In context to existing work, the study presented in this paper makes the following novel research contributions:

1. A survey of Software Maintenance professionals from four popular open-source projects (Google Chromium, Apache OpenOffice, Seamonkey and Mozilla Thunderbird) on the topic of web-link (references to knowledge sources) sharing in issue tracking discussion forum (refer to Section 6.2).

2. An in-depth characterization study on URLs and Web-links in Google Chromium Issue Tracking discussion forums. We frame several research questions and conduct a series of

experiments (on open-source publicly available dataset) on more than 200000 issue reports (4 years and 7 months of dataset) to answer the stated questions (refer to Section 6.3).

3. A Google Chromium browser extension (called as *Samekana*) to help bug reporters, bug triagers, bug fixers and quality assurance managers in saving, organizing and including references to knowledge sources in issue tracking system comment post. The browser extension is freely and publicly available on Chromium Webstore (refer to Section 6.4).

6.2 Survey of Software Maintenance Professionals

The research work presented in this paper is motivated by the need to investigate solutions to common problems encountered by Software Maintenance Professionals. We believe that inputs from practitioners are needed to inform our research and hence we conduct a survey of developers from four open-source projects. We sent an email with the link to our survey form to the project mailing lists (for example: chromium-dev@chromium.org) of Google Chromium[1], Apache OpenOffice[2], Seamonkey[3] and Mozilla Thunderbird[4] projects. We received a total of 14 responses (7 from Google Chromium developers, 2 from Apache OpenOffice developers, 2 from Seamonkey developers and 3 from Mozilla Thunderbird developers).

Table 6.1 shows the result of the survey consisting of 6 mandatory questions. The results of the survey (refer to Question 4 in Table 6.1) reveals that 78% of the developers believe that inclusion of links to references (and relevant knowledge-sources on the Internet) improves (some improvement to large improvement) the quality of bug report and comments posted as solutions to a reported bug. Developer responses to Question 6 shows that **71% of developers see some value to high value** (in-terms of enhancing the quality of comments posted in online issue tracking systems) in a browser or an issue tracking system plug-in which makes it easy to include references or citations to relevant web-pages (in comments) encountered while doing a web-search before reporting a bug or posting a solution in response to a bug report. Responses to Question 1, 2, 3 and 5 of Table 6.1 reveals that as bug reports and bug fixers conducting an extensive online search (while reporting and fixing bugs) and including links (URLs) to relevant knowledge-sources (relevant references and web-pages on the Internet) is an important and common activity.

[1] http://www.chromium.org/
[2] http://www.openoffice.org/
[3] http://www.seamonkey-project.org/
[4] https://www.mozilla.org/en-US/thunderbird/

6.3 Characterization Study

Experimental Dataset

We conduct experiments on dataset downloaded from the issue tracking system of Google Chromium Browser project[5]. Google Chromium Browser is a large, long-lived and complex software open source project. Google Chromium Browser issue reports are publicly available and hence the experimental analysis presented in this paper can be replicated by other researchers and used for benchmarking and comparison. We used Google Chrome issue tracker API[6] to download the dataset.

Table 6.2 shows the experimental dataset details. As shown in Table 6.2, we download 4 years and 7 months of issue reports (and all the comments in response to the issue reports until April 2013). We download the issue report as well as comments, labels and issue report fields. As shown in the Table 6.2 we downloaded 207185 issues having 1818790 comments. The number of issues in the experimental dataset which are labeled as *closed* is 162665 and number of issues marked as *open* is 44520.

We extract all the links (URLs) present in the issues using regular expressions and string pattern matching. We extract links from issues description (issue or bug report) and issue comments. We observe presence of many spurious links present in the issue tracking system (URLs such as: www.foo.bar.com and www.blah.blah.com) and filtered (data cleaning before mining) such links as they are not relevant (noise) with respect to the analysis done in this study. After removal of spurious links, we get 113176 links from issue description and 1005928 links from issue comments (refer to Table 6.2). The number of URLs present in issue report and comments shows that linking knowledge sources and relevant information present in external sites and applications is a common phenomenon. Table 6.3 shows distribution of issue reports across total number of links present in issue description and comment. The result in Table 6.3 reveals that more than 30% of issue reports have at-least one URL in the issue description itself and **more than 60% have at-least one link in either the issue description or comment**. Empirical results (refer to Table 6.3) indicate that more than 10% of issue reports contain more than 2 links in the description itself and **more than 10% of issue reports contain more than 10 URLs in the description or comments**.

Domain Categorization and Frequency

We extract domain names (such as chromium.org or google.com) from all the URLs (embedded in issue report description and comment) in the experimental dataset and identify the most frequent domains and domain types. We notice that there are 12248 distinct domains in issue descriptions while there are 9217 distinct domains present in issue comments. Table 6.4 reveals Top 20 domain names and their frequency counts present in issue report description and comments respectively. We observe that the **Top** 20 **domains do not belong to any one or two specific categories and is rather a combination of multiple categories**. For example, we notice large number of links to

[5] https://code.google.com/p/chromium/issues/

[6] https://code.google.com/p/support/wiki/IssueTrackerAPI

websites from software vendors (Microsoft.com), social networking sites (Facebook.com), video sharing sites (Youtube.com), code development site (github.io), internal developer repositories (go, crash), sites such as jsFiddle which allows developers to simulate Ajax calls, knowledge sources as Wikipedia and reference websites such as W3.org. We notice that few links present in the Top 20 domains in description and comments shows similar distribution (such as google.com, chromium.org and appspot.com, jsFiddle.com, Wikipedia.org, github.io and w3.org are prevalent both in comments and description). We observe that few domains are in Top 20 of description (facebook.com) and not in comments and vice-versa (blogspot.com).

To gain a deeper understanding of various domains and types of domains present in the dataset, we conduct an analysis of Top 100 domains present in all the URLs extracted from bug description. We define 13 mutually exclusive categories (such as search engine, software download website, code development and Q&A website for programmers such as jsFiddle and Stackoverflow.com respectively: refer to Table 6.5) and classify each domain (only Top 100 domains) to one of the 13 predefined categories. We apply the rule that if a domain is not found to be suitable to any pre-defined category then we assign it to a category called as *other*. Table 6.5 shows various categories, domain categorization and link count (in terms of %) for each of the pre-defined categories for bug description and comments respectively. Table 6.5 shows that **Top 100 domains contribute to only 70%** of all links present in the issue descriptions (there is a large variability in the domain types and domains). In contrast, the Top 100 contribute **92% of all links present in comments**. Experimental results in Table 6.5 reveals that links to search engines, software repositories used by the Chromium developer community and code development related as well as reference sites are among the most popular domain or link types in the issue tracker. We found a similar pattern and results for Top 100 domains of comments (due to space constraint in the paper detailed results are not presented for the comments and are provided only for the issue description)

Link Characterization for Different Issue Type

The issue tracker provides bug reporters, triagers and bug fixers a functionality to assign labels to issues. Issues can be tagged according to their types such as: Security, Performance, Crash, Polish and Clean-up. We extract labels from issue reports and conduct a study to understand the similarities and differences in the characteristics of different issue types with respect to the frequency and types of URLs. Table 6.6 shows the number of issues (in the experimental dataset) labeled with six different types of issues. We observe that not every issue report is labeled and hence our analysis is limited to the issues for which a label is assigned.

Table 6.6 reveals that there are more than 10000 issues labeled as *Crash*. The number of issues with Crash label is highest among all the categories followed by *Regression* and *Security*. Table 6.6 and 6.7 shows frequency count of issue report (and other statistics) containing links in issue description and issue comments respectively. We compute the average number of links with respect to each issue description and comments (refer to Table 6.6 and 6.7). Experimental results in Table 6.6 reveals that performance and crash issue types have the highest number of links present in the issue description. We conduct a manual analysis and visual inspection of the links. We observe the presence of specific types of links in crash and performance issue description : **57% of links**

in crash issue points to local repositories of developers (refer Table 6.6). *Local repositories* are the folders/links that are accessible only to developers, like *Go, crash etc.* Performance bug description consists of more than **45% links to *performance* files/links**. Manual analysis of these links reveals that these are special links used to display performance of various components, and hence present in most of performance issues.

We count links present in the comments of each of the five issue type. Our findings show that **86% of the security issues have links in comment**, which is highest among all the categories (not considering polish issue type because of insignificant number of issues). This is also much higher than security issues having links in description. Manual analysis of security issue comments indicates two possible reasons for presence of large number of links is security issue comments. The first reason is: **99% of the security issues are marked as closed** and hence there is much higher probability of developers sharing links to source-code files in security issue comments as compared to other categories (refer to Table 6.7). The second reason is that developers frequently share links to *t*est files in security issues. Nearly **13% of all the links in security issue comments are links to *test* files or folders** as shown in Table 6.7. This shows the practice of developers towards appropriate testing of security issues.

Correlation of Link Count with Comment Count and Bug Fixing Time

We hypothesize that large number of links present in an issue can be an indication of increased discussion among developers and hence can increase bug fixing time. To test both the hypothesis we calculate correlation of total links present in issue description and comment with number of comments present in issue and issue fixing duration. Correlation between number of comment and number of links is 0.52, relatively higher than the correlation between number of links and bug fixing time which is 0.18. Figure 6.1 shows scatter plot of average issue fixing time of a bug report varying with the number of links present in the issue description and issue comment. Similarly Figure 6.2 shows scatter plot of average number of comments varying with the number of links present in the issue description and issue comment. Figure 6.1 and Figure 6.2 also plots the Bzier curve approximation of the points. It is interesting to note that **average issue fixing time and average numbers of comments are slightly increasing with number of link present** in the bug report.

Code Development Trend Analysis

Table 6.4 shows that developers use different types of online environment like github, jfiddle and html5rocks etc. As shown in Table 6.5, they constitute 2.28% and 0.86% of all links present in Chrome issue tracker decription and comment respectively. To understand how developers (or users) preferences of code developent environment have changed with respet to chome issue tracker, we perform trend analysis (programmatically) of 6 most popular code develpemt sites. Popularity is measured by the number of links present in Chrome issue tracker, more links, more popular. googlecode.com, jsfiddle.net, jsbin.com, htmel5rocks.com, github.io and khronos.org are the six sites we analyzed. Our dataset have a span of nearly 4.8 years (Aug-2008 to Apr-2013). We

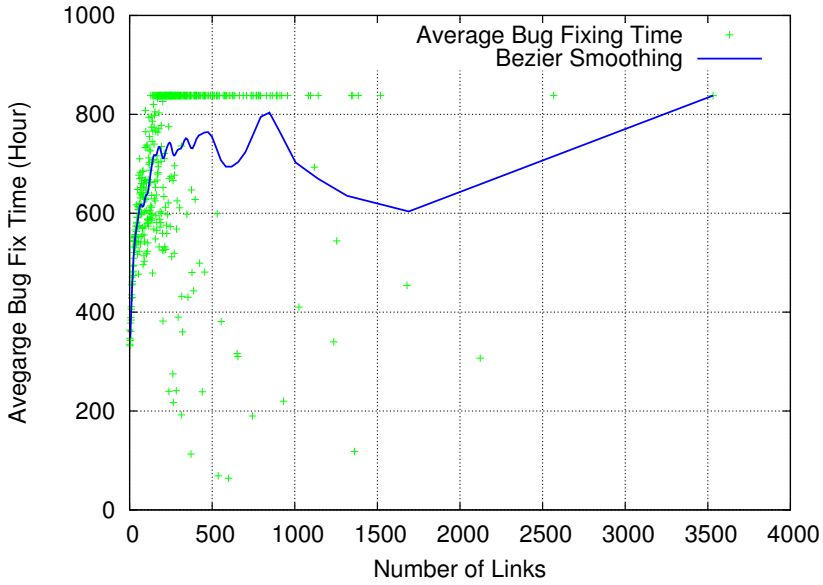


Figure 6.1: Scatter plot of average bug fixing time with number of links

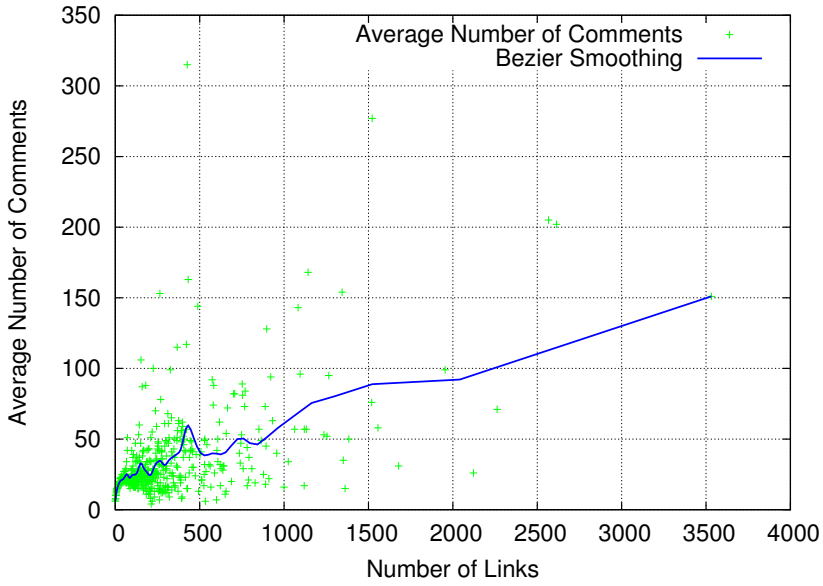


Figure 6.2: Scatter plot of average number of comments with number of links

count for each of the six sites number of links shared by developers in each year. Figure 6.3 and Figure 6.4 shows trend line of each site for description and comment separately. It is interesting to see increase in popularity of "jsfiddle" and "github" code development environment. In 2012, **814 and 481 unique links of jsfiddle and github respectively, are shared in chrome issue tracker**. There is also slight increase in the trend line of other code development sites, refer Figure 6.3 and Figure 6.4. *jsfiddle* is a code development environment that allows users to write and execute

JavaScript code. Gomez et al. reported similar finding on StackOverflow data, where *jsfiddle* was found to be one of the top 3 websites (in terms of unique links shared on StackOverflow) [64]. Trend analysis of code developement sites can be useful for the developers to know about popular tools and libraries.

Presence of Links to Image Sharing Website in Issue Tracker System

As shown above in Table 6.5 developers frequently share links to image sharing website like tinypic, flicker, imageshack etc. It is not known why developers tend to share links of image sharing website or how they are helpful in issue fixing. Manual analysis of 50 random issues report shows that developer share links to this website to share screenshots of problems, as only 7 out of 50 issue reports were about problems related with image sharing website. Discussion with some of chromium developers revels that **links to image sharing website are not much popular among chromium developers** as these links become invalid after some time and information about the link is lost. Following is a comment of one of the chromium developer:

"*links to outside websites eventually stop working (image removed, bandwidth exhausted, etc). A screenshot attached directly to the report is much more useful. Same thing with log files and everything else"*.

Above result can be helpful to guide and motivating users to upload links of image sharing websites where content is not lost. Tools support can be provided to users to automatically fetch images from such link and upload them to issue tracker system.

Link Characterization with Authors, priority and Severity

There are 46960 distinct authors present in our dataset. 19262 authors reported issues consisting of at least one link in description. As shown in the Table 6.8 14325 authors reported at least one issues having link. Also there are **more than 1000 authors who reported more than 6 issue reports consisting of links**. This shows that posting of web-links to knowledge resources is a practice followed by several developers and strengthens our motivation to develop a tool like Samekana. Our goal is make it simple for developers to include links/references in issue tracker system.

Chrome issue tracker allows five level of priority $(0-4)$, 0 being highest priority and 4 being lowest. Similarity Chrome issue tracker allows 4 level of severity $(0-3)$, 0 being most sever. 203563 and 19845 issue had priority and severity label in our dataset. We observe slight increase in presence of links for high priority and high severity issue reports. As shown in Table 6.9 34% of priority-0 and 35% priority-1 issue reports consists of links, which is marginally higher from priority-2 (32%) and significantly higher from priority-3 (25%) issue reports. Similarly, 35% of severity-0 and 31% of severity-1 issue reports have links. We observe increasing trend of links and severity, percentage of issue reports with links are increasing with increase in severity. This shows **correlation between priority or severity of issue report and presence of link** in it, refer to Table 6.9.

6.4 Samekana Browser Extension

We developed a Chrome Browser Extension[7] called as Samekana (programs that can extend and modify the functionality of the Chrome Browser) and made it publicly available through the Google Chromium Webstore extension and application distribution marketplace. The browser extension is freely available for anyone to download and use and currently works on Google Issue Tracker for the Chromium and Android Projects. The short URL to Samekana is goo.gl/WvLuF or one can go to the Chrome Webstore [8] and search the store with the Keyword Samekana. The browser extension can be installed by clicking the "Add to Chrome" button. Following are the main features of Samekana.

1. *Save* and *Annotate* web-references (while referring to knowledge sources on the Web) relevant to an issue report
2. *Organize* and *Search* web-references using *Tags* and *Labels*
3. *Include* web-references (individual or all labeled with a pre-defined tag) in Issue Tracker *Comment Posts*
4. *Include* web-references from *Browsing History* in Issue Tracker comment Posts

Figures 6.5, 6.6 and 6.7 are snapshot of Samekana browser extension demonstrating the features and use-case scenario. Figure 6.5 displays the snapshot of Samekana explaining the annotate web-reference using tags functionality (window in Figure 6.5 can be launched by clicking on the save current web-page as a reference button on Samekana main page). Label *A* in Figure 6.5 shows the text-box the entering the title of the reference (which appears in the reference list of the discussion forum comment box along with the URL). Label *B* shows the text-box for description (additional notes and comments to the title) and Label *C* shows the text-box for specifying Tags. Tags are labels which are user-defined used to organize and structure web-references within Samekana (multiple tags can be specified for a reference). As shown in Figure 6.5, once can assign the issue report ID as the tag (in order to organize all web-references related to a specific issue report in one folder).

Figure 6.6 shows a snapshot of Samekana demonstrating the functionality of editing, deleting, searching and adding individual or all references (for a pre-defined tag) to the issue tracker comment post. Label *D* in Figure 6.6 shows the All button for displaying the browsing history (so that additional references not annotated explicitly can be included if required). A user has the option to enable or disable reading the browsing history (privacy setting). A user can disable reading of browsing history and save web-references which are explicitly annotated. Label *E* shows the search box which can retrieve past references based on simple keyword based search. Label *F* shows previously saved references for a particular tag (selected by the user). As shown in the Figure 6.6, a user can edit or delete any saved reference.

[7] http://developer.chrome.com/extensions
[8] https://chrome.google.com/webstore

Figure 6.7 shows the snapshot of the discussion form comment box of Google Chromium and Samekana. As shown in Figure 6.7, Samekana browser extension inserts an icon and a message with the directions while loading the page for an issue on the issue tracker (refer to Label *G* of Figure 6.7). Label *H* of Figure 6.7 shows the list of references (organized like a bibliography which can be cited similar to in a technical or research paper) included in the comment box as a result of clicking the include reference link (refer to Label *I*).

6.5 Limitations and Conclusion

We conduct a survey of software maintenance professionals from four open-source projects and asked questions to the core developers of the respective projects. However the number of survey respondents were limited and equal to 14 (due to average response rate) and more responses can increase the strength of the conclusions from survey results. We conduct empirical analysis (link characterization study) on dataset belonging to one project (Google Chromium Browser) and hence one threat to validity of our work is that the results can be biased to the characteristic of one project.

The survey response of Software Maintenance professionals from four open-source projects reveal that majority of bug reporters and bug fixers conduct web search before posting issue reports and their comments. Survey response indicate that presence of web-links (in issue tracker discussion forum) as citations to knowledge-sources and artifacts are useful and there is value in brining tool support for software maintenance professionals which can facilitate easy saving, organizing and posting of web-links encountered during bug resolution. A characterization study of web-links shows a large variability in terms of the websites referred through links in issue tracking discussion form. The characterization study reveals patterns which we believe are useful to the practitioners community. We develop a Chromium Browser extension as tool support (publicly available and freely downloadable) to developers for saving, organizing and posting web-links.

Table 6.1: Survey Results [14 responses]: Chromium, OpenOffice, Seamonkey and Thunderbird developers on Web Link (References on Internet) Sharing in Issue Tracking System Discussion Forum

Q1: As a Bug Reporter do you include links (URLs) to relevant knowledge-sources (relevant references and web-pages on the Internet) in the bug report?	
Not much	28.57%
Sometimes	35.71%
Most of the time	35.71%
Q2: As a Bug Reporter do you conduct an extensive online search before submitting a bug report?	
Not much	35.71%
Sometimes	14.28%
Most of the time	50.00%
Q3: Do you include links (URLs) to relevant knowledge-sources (relevant references and web-pages on the Internet) when you post solutions in response to bug reports?	
Not much	28.57%
Sometimes	57.14%
Most of the time	14.28%
Q4: Do you believe that inclusion of links to references (and relevant knowledge-sources on the Internet) improves the quality of bug report and comments posted as solutions to a reported bug?	
Not much	21.42%
Some improvement	35.71%
Very much	42.85%
Q5:As a Bug Fixer or Developer do you conduct an extensive online search before posting solutions in response to bug reports?	
Not much	35.71%
Sometimes	28.57%
Most of the time	35.71%
Q6:Do you see a value (in-terms of enhancing the quality of comments posted in online issue tracking systems) in a browser or an issue tracking system plug-in which makes it easy to include references or citations to relevant web-pages (in comments) encountered while doing a web-search before reporting a bug or posting a solution in response to a bug report?	
Not much value	28.57%
Some value	64.28%
High value	07.14%

Table 6.2: Google Chrome browser project experimental dataset details

Field	**Value**
First Issue ID	2
Last Issue ID	237020
Reporting Date of First Issue	30-08-2008
Reporting Date of Last Issue	30-04-2013
Issues Downloaded	207185
Number of Issues restricted/permission denied	29834
Number of Open Issues	44520
Number of Closed Issues	162665
Number of Comments	1818790
Number of links in Issue description	113176
Number of links in Issue comment	1005928

Table 6.3: Distribution of Issue reports based on number of links in description and comment, Desc: Description, Comm: Comment

No of links (Desc)	Count (%)	No of links (Desc+Comm)	Count (%)
1	43680 (21.08)	[1-10]	106837 (51.56)
2	14817 (7.15)	[11-20]	11364 (5.48)
3	4859 (2.34)	[21-30]	3826 (1.85)
4	1612 (0.78)	[31-40]	1774 (0.86)
5	743 (0.36)	[41-100]	2915 (1.41)
6	382 (0.18)	[101-200]	751 (0.36)
>= 7	811 (0.39)	> 200	385 (0.18)
Total	66904 (32.29)	Total	127852 (61.7)

Table 6.4: Top 20 domains and their link count in bug description and bug comment, Desc: Description, Comm: Comment

Desc	Count	Comm	Count
google.com	25545	**chromium.org**	836395
chromium.org	16180	**google.com**	49597
webkit.org	8918	appspot.com	42251
crash	4543	webkit.org	21939
go	3478	crash	8256
appspot.com	2597	go	1769
w3.org	1659	w3.org	1359
youtube.com	1564	mozilla.org	1238
jsfiddle.net	1430	microsoft	1160
cautotest	843	youtube.com	1042
microsoft.com	760	cautotest	954
chrome.com	726	googleapis.com	911
wikipedia.org	712	github.io	849
github.io	688	chrome.com	809
facebook.com	668	googlecode.com	692
dzview.com	644	blogspot.com	688
mozilla.org	619	jsfiddle.net	645
chromegw	570	wikipedia.org	623
googlecode.com	456	chromegw	538
yahoo.com	438	apple.com	528

Table 6.5: Categorization of Top 100 domains, Desc: Description, Comm: Comment

Category	Domain Name	Count (Desc)	%Total Links	Count (Comm)	%Total Links
Search Engine	Google, Yahoo, bing, yandex, msn, app, baidu	26347	**23.28**	49980	**4.97**
Devlopers Repository	Go, Crash, chromegw, chromebot, issue, folder, crash-staging, goto, cautotest	9803	8.66	12082	1.2
Video Sharing/Image Sharing	Youtube, imgur, imageshack, screencast, vimeo, dropbox, tinypic, flickr, mtv, hulu, cl	2583	**2.28**	2138	**0.21**
Social Network	facebook, twitter, sina, myspace	1044	0.92	520	0.05
E-commerce	Amazon, ebay	281	0.25	141	0.01
S/W Companies	Microsoft,apple, adobe, oracle	1512	1.34	2090	0.21
Wiki	Wikipedia, wikimedia	878	0.78	706	0.07
Code development	Satckoverflow, jsfidlet, github, google-code.com, jsbin, khronos, html5rocks, whatwg, chromeexperiments, pastebin, codepen, jsperf, goo, freedektop, ,w3schools, msdn	2578	**2.28**	2661	**0.26**
News	cnn, bbc, nytimes, wheather	513	0.45	285	0.03
Other	Chromium, webkit, appspot.com, chrome, mozillla, googleapis.com, angrybirds, atdmt, launchpad, s3.amazonaws.com, bt5156, ipark, greenbytes, popuptest, macromedia, live, acidtests, dzview, dropboxusercontent, xkcd	30612	**27.05**	903296	**89.8**
Licensing	W3, iana, gnu, ietf, irs	2193	1.94	1885	0.19
S/W Download	sourceforge, cnet, java, kernel	388	0.34	464	0.05
blogs	Wordpress, googleusercontent, tumblr, blogspot	537	0.47	878	0.09
OS	Genome, ubuntu, gentoo, android	222	0.2	588	0.06
	Total=	79491	**70.24**	977714	**97.2**

Table 6.6: Link distribution in different bug type (Description), Desc: Description, Avg: Average, Perf: Performance files

Issue Type	No of Issues	Link in Desc (%)	Total links	Avg. Link (Per Desc)	Local Repos. (Desc)(%)	Perf. files (Desc) (%)
Crash	10020	**5972** (59.6)	9676	0.96	**5542 (57.28)**	11 (0.11)
Security	2381	1259 (52.88)	2651	1.11	115 (4.34)	7 (0.26)
Regression	9232	3596 (38.95)	6421	0.69	972 (15.14)	666 (10.37)
Performance	1434	**870** (60.67)	1841	1.28	3 (0.16)	**836 (45.41)**
Polish	20	1 (5)	1	0.05	1 (100)	0 (0)
Cleanup	49	10 (20.41)	14	0.28	1 (7.14)	0 (0)

Table 6.7: Link distribution in different bug type (Comment), Comm: Comment

Issue Type	No of Issues	Link in Comm (%)	Total links	Avg. Links (Per Comm)	Total Comments	Link to 'test' Files (%)	Fixed Issues (%)
Crash	10020	6321 (63.08)	43812	6.93	106292	3466 (7.91)	8301 (82.84)
Security	2381	**2056 (86.35)**	20169	9.81	52801	**2665 (13.21)**	**2363 (99.24)**
Regression	9232	**6545 (70.89)**	50290	7.68	130386	3946 (7.85)	8305 (89.96)
Performance	1434	944 (65.83)	10984	11.63	15520	1270 (11.56)	880 (61.37)
Polish	20	18 (90)	143	7.94	287	0 (0)	19 (95)
Cleanup	49	30 (61.22)	3375	112.5	494	547 (16.21)	25 (51.02)

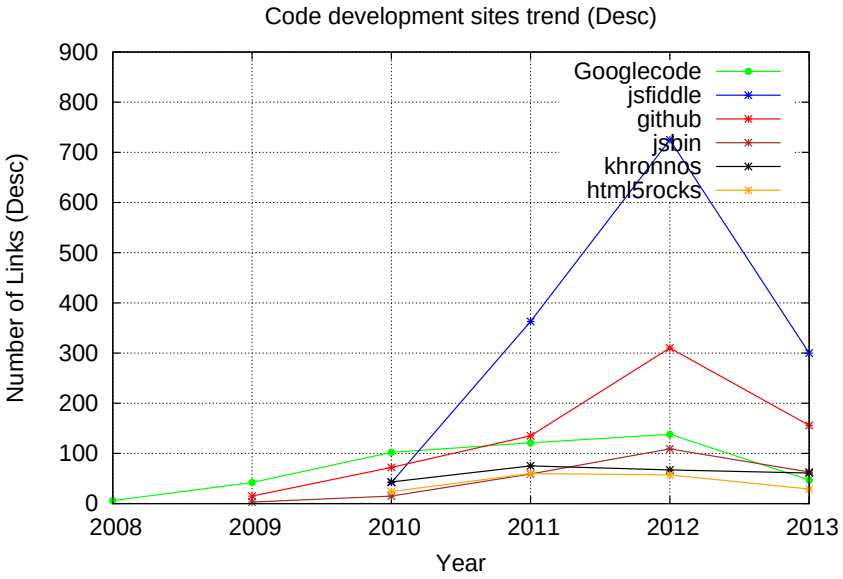


Figure 6.3: Link sharing trend of code development site (Description), Desc: Description

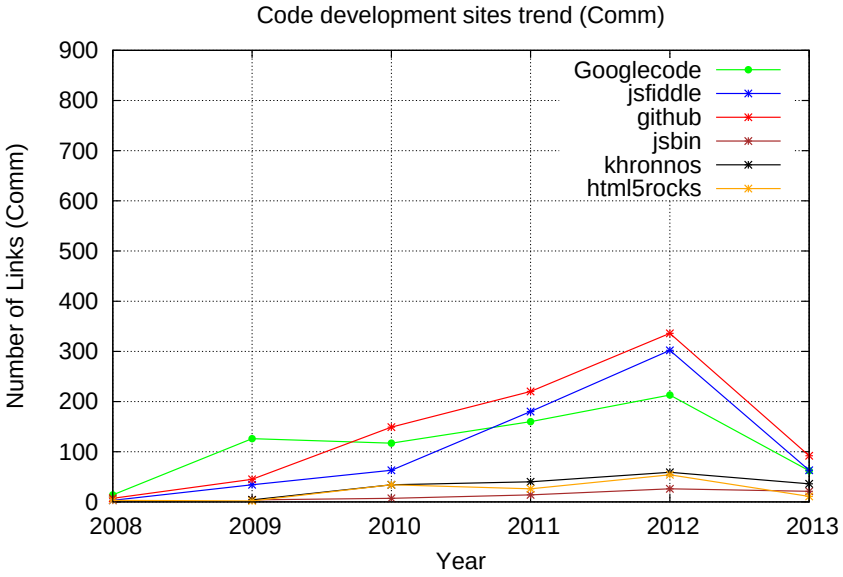


Figure 6.4: Link sharing trend of code development sites (Comment), Comm: Comment

Table 6.8: Authors and link distribution (Description)

Issues	No. of authors
1	14325
2	2282
3	803
4	381
5	243
6	147
>= 7	**1081**

Table 6.9: link distribution With Respect to Priority and Severity of Issue Report (Description), Pri: Priority, Sev: Severity

Pri.	Issue Count	Issues Containing Link (%)	Sev.	Issues Count	Issues Containing Link (%)
0	3042	**1064 (34.98)**	0	832	**295 (35.46)**
1	31686	**11139 (35.15)**	1	4985	**1588 (31.85)**
2	156754	50464 (32.19)	2	12150	2833 (23.32)
3	12079	3071 (25.42)	3	1878	388 (20.66)
4	2	0 (0)	-	-	-
Total	203563	65738 (32.29)	Total	19845	5104 (25.72)

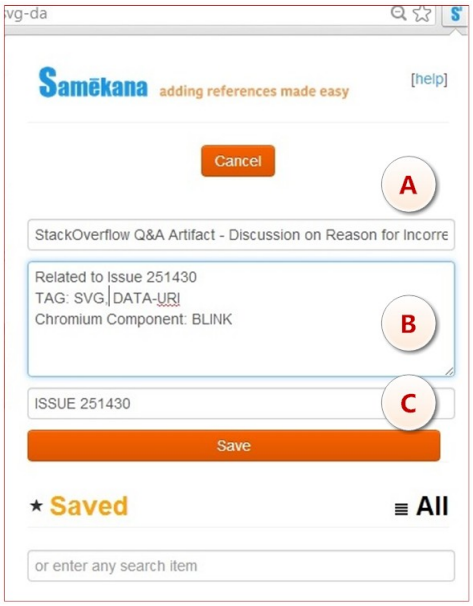


Figure 6.5: A snapshot of Samekana Annotate and Save Reference Feature. Labels: (A) Title, (B) Description and (C) User Defined Tag

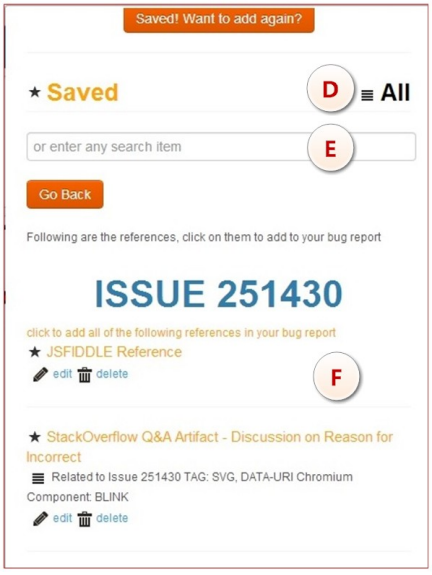


Figure 6.6: A snapshot of Samekana View, Edit, Delete and Post Reference Feature. Labels: (D) View Browsing History, (E) Search References (F) Edit, Delete and Post References Tag

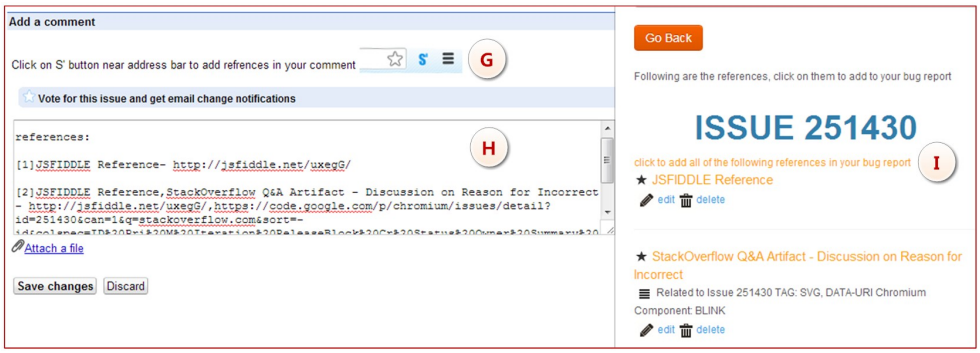


Figure 6.7: A snapshot of Issue Tracker Comment Box and Samekana Post Reference Feature. Labels: (G) Samekana Label in Issue Tracker, (H) Posted References in Comment Box (I) Samekana Link to Include References

Chapter 7

Interaction Based Topic Centric Community Discovery on Twitter

In the previous two chapters, we focused on systems to enhance content quality. Now, we look at the third aspect of content quality on Web 2.0 services – Information Retrieval Enhancement. Here, we exploit quality content in social media to aid effective Information Retrieval. In this chapter, we exploit interactions between individuals on Twitter to discover topic-centric homogeneous communities.

7.1 Research Motivation and Aim

Over the past decade, there has been a swift rise in the number of users who register on online social networking websites like MySpace, Facebook, and YouTube. Registered users on these social networks are provided with various features like reply, comment, subscribe and friendship in order to interact, engage and share information with each other. Such interactions lead to the formation of closely knit user-groups or densely connected clusters of users around specific topics within the social network; these are called communities. The tendency of users to form a community structure is a significant characteristic of any social network. *Community discovery* or *Community detection* in social networks has many practical applications and hence, is of research interest to both physicists and computer scientists alike. However, online social networks like Facebook and YouTube have a large user base and host enormous amounts of data. For example, YouTube has more than 800 Million unique visitors per month, 100 Million social interactions per week and more than 48 hours of video is uploaded every minute.[1] Due to the existence of such large scale networks and huge volume of data, community discovery on social networks is a challenging information extraction (or retrieval) task.

Currently, Twitter is one of the most used and immensely popular social network. Twitter is a micro-blog which allows registered users to share images, videos and text in short 140-character limit messages called *tweets*. Twitter reports that it has more than 100 Million active users with

[1] `http://www.youtube.com/t/press_statistics`

more than 200 Million tweets posted everyday. Previous research shows that Twitterers use Twitter to serve multiple purposes like - share their daily experiences, take part in conversations, share information (in the form of URLs, images, videos) and report & assimilate news [83]. Hence, Twitterers with homogenous interests tend to *flock together* viz. form cohesive self-formed communities. Moreover, these communities rally around specific topics which are of prime interest to Twitterers part of that specific group or community [81]. For example, fans of the well known American singer Beyonce may form a community around the topic 'Beyonce' while fans of Lionel Messi (Argentina & Barcelona football player) may form a community around the the topic 'Messi'. Extraction or discovery of these tight-knit homogeneous communities on Twitter around specific topics of interest has wide spread real-life applications across multiple domains. We present two real-word examples to demonstrate our argument :

1. **Digital Marketing –** Many businesses use Twitter as a marketing tool and implement roadmaps depending on their Twitter connections [84]. Previous studies have also shown knowledge of topic-specific communities and it's key players can aid businesses to efficiently advertise and market their products [72, 79]. Therefore, the detection of topic-centric communities can assist digital marketing teams to identify targeted consumer interest groups on Twitter. Such information could be highly valuable to plan product releases and announcements in context with the needs of the customer.

2. **Law Enforcement Agencies –** Twitter is a vibrant platform to engage and share opinions on varied topics. Twitter like other Web 2.0 technologies also has various advantages like low publication barrier, anonymity and ease of access. However, these advantages are exploited by malicious users for malignant activities like cyber activism and promotion of hate speech.[2] Previous studies also show that social media is used by malicious users for hate speech and propaganda [118]. Hence, it has now become paramount for law enforcement agencies across the world to extract topic-centric communities on Twitter around sensitive topics and identify the key actors within those communities.

7.2 Technical Challenges

1. **Large Scale Network –** Twitter is a large scale social network consisting of more than 100 Million registered users. The size of topic-centric communities are much smaller than the size of the overall network. Thus, discovering topic-centric communities in such large scale networks is equivalent to *finding a needle in a haystack*. Previous research in detecting communities in social media require complete knowledge of the nodes (users) in the social network [104, 108, 109, 133]. Hence, these methods may not scale to the ever increasing sizes of current social networks like Twitter. A few solutions in literature have proposed algorithms for large scale networks but is still an area of active research [102, 130].

[2] `http://www.witness.co.za/index.php?showcontent&global[_id]=74740`

2. **Dynamic Nature –** More than 200 Million tweets are posted on Twitter everyday and more than 400,000 new Twitter accounts are created per month.[3] Such dynamic nature leads to rapid changes in formation, evolution and termination of communities. Some approaches in literature address these issues by taking the dynamic nature of the social network into account [36, 120, 122, 132].

3. **Heterogenous Interactions –** Twitter, akin to other social networks like Facebook, provides its users with multiple ways to interact with each other. These interactions include follow, reply, mention, retweet and favorite. Due to presence of such multiple interactions, determining user community association is difficult. In addition, combining all interactions into one may lead to noisy results as certain interactions may statistically dominate others and may not reveal the true underlying social structure. Mining heterogenous networks for community discovery has been relatively less explored in literature and is an open research problem [120, 121].

4. **Community Evaluation –** A community has no quantitative definition and hence, is subjective and open to interpretation. The most conservative definition of a community requires all nodes in the community to be connected to each other. However, real-life social networks like WWW don't conform to such conservative definitions [82]. Moreover, it's not possible to have a ground truth of the community structures in large scale social networks. Therefore, there's a need to evaluate communities in such networks which are devoid of ground truth. Also, topic-centric communities need to be semantically evaluated with respect to common topic shared by the community [119]. Various network-based community quality functions like *Modularity* have been proposed in literature to measure the quality of a community [98]. Evaluating communities on large scale networks is still a challenge, nonetheless.

The specific research aim of this paper is to investigate the use of local information algorithms to discover and analyze topic-centric interaction based communities on Twitter to aid real-world applications like Digital Marketing and Law Enforcement Agencies.

7.3 Research Contributions

1. *First focussed study on discovering interaction-based topic-centric communities on Twitter.* To the best of of our knowledge, this is the first empirical study to discover interaction-based topic-centric communities in Twitter. We use a publicly available Twitter dataset accessible using a web service and perform empirical analysis on two diverse topics – *CAD* (Computer Aided Design) and *Kashmir* (a disputed region in the northern part of India). We evaluate the output communities along multiple dimensions to assess the network structure quality as well as the network semantic quality; which helps throw light on the efficiency of our approach.

[3] http://blog.twitter.com/2011/03/numbers.html

2. *Investigation of a community detection algorithm relying on local information to infer topic-centric community structures on Twitter* – The application of the local information algorithm based on *Local Modularity* (LM) to discover topic-centric communities is a unique contribution in context to previous approaches [104, 109, 128, 133]. The LM algorithm embodies a formal mathematical proof to extract concentric communities around specified nodes within a network [50]. It also entails the use of an interactive procedure guided by an objective function to detect topic-specific communities from local signals. The LM algorithm aims to maximize the structural quality of the community by optimizing a pre-defined objective function which calculates the density of edges of nodes within a community. The objective function viz. *local modularity* is based on the structural properties of the network graph based on user interactions.

3. *Use of heterogenous interactions to represent connections between users* – We represent Twitter as a weighted directed graph where each node signifies a Twitter user and each weighted edge signifies the total weight of the interaction such as reply, mention or retweet between the connected nodes. We make use of a publicly available Twitter graph metric which combines heterogenous interactions like mention, reply and retweet between Twitterers . The weight of the directed edge between two users is directly proportional to the magnitude of interactivity (preserving the direction). For example, a directed edge from user X to user Y with weight w indicates that user X interacts (replies, mentions, retweets) with user Y on a strength equivalent to the value of w. The greater the value of w, the greater the user X interacts with user Y.

7.4 Solution Approach

Twargon – Twitter jargon

Twitter provides many features to registered users to engage in multiple types of interactions with each other. Here, we briefly mention these features.

1. *Follow* – Twitter users connect with each other via a subscription based feature called the *Follow*. Twitter users can choose to 'follow' other fellow users to receive their tweets. For example, a user X can receive tweets from user Y if X 'follows' Y, indicating information flow from Y to X. The users who 'follow' user X are called *followers* of X and the users whom X follows are called *followees*.

2. *@-messages* – Twitter provides a feature to reply as well as send tweets to other users called *@-messages*. For example, a user X can reply or send a tweet to another user Y by writing @Y (where Y is a username) indicating an information flow from X to Y. Moreover, a user X can send an @-message to any other public user Z without *following* Z. An important distinction to note here is that the direction of information flow using *@-messages* is opposite to that of *Follow*.

3. *Retweet or RTs* – Twitter allows its users to repost a tweet using the *ReTweet* (RT) button or by copying the previous message and prefixing the characters 'RT' to the tweet. An RT by a user signifies that the tweet is *interesting* to the user and hence, wants to re-share it. Similar to @-messages, a user need not follow another public user to RT his tweets.

4. *Lists* – Twitter users can categorize other users into buckets called *lists*. Users need not follow other users to create lists. Lists provide a great way to organize groups of users connected via a common theme. Lists can be publicly accessible to everyone or private to the owner.

Problem Statement

We represent Twitter as a directed weighted graph $\mathcal{G} = (V, E, W)$. $V = \{v_1 \dots v_m\}$ is the vertex set where v_i denotes a Twitter user. E is the edge set where e_{xy} denotes a directed edge from v_x to v_y where $v_x, v_y \in V$. W is the set of weights where w_{xy} is weight of the edge e_{xy} and $w_{xy} \in \mathbb{R}$. Let $\mathcal{I}(RT_{xy}, @_{xy})$ denote the strength of interaction from user v_x to v_y based on @ – *replies, mentions* and RT – *retweet*. We create an edge e_{xy} iff $\mathcal{I}(RT_{xy}, @_{xy}) \neq 0$ and we assign $w_{xy} = \mathcal{I}(RT_{xy}, @_{xy})$. Given a list of topics $\mathcal{T} = \{t_i \dots t_n\}$ and $\mathcal{G} = (V, E, W)$, our goal is to find the set of topic-based communities $C = \{C^{t_1} \dots C^{t_n}\}$ where $C^{t_i} \in \mathcal{G}$ is the Twitter community for topic t_i. The notations used in the algorithm are described in Table 7.1.

Notations	Description
$\mathcal{G} = (V, E)$	Complete Graph
$\mathcal{T} = \{t_1 \dots t_n\}$	Set of n topics
$C^{t_i}_{final}$	Final community for topic t_i
$C^{t_i}_{current}$	Current community for topic t_i
$\mathcal{G}^{t_i}_{visited}$ $(V^{t_i}_{visited}, E^{t_i}_{visited})$	Final visited graph for topic t_i
$\mathcal{G}^{t_i}_{current} =$ $(V^{t_i}_{current}, E^{t_i}_{current})$	Current visited graph for topic t_i
$\mathcal{U}^{t_i}$	Undiscovered nodes for topic t_i
$\mathcal{B}^{t_i}$	Boundary nodes for topic t_i
$\mathcal{R}^{t_i}_C$	Local Modularity of C for topic t_i
$\mathcal{R}_{v_i} : v_i \in \mathcal{U}^{t_i}$	$\mathcal{R}$ for community when v_i added to $C^{t_i}_{current}$
$\mathcal{R}^{v_i}_{max}$	$\mathcal{R}$ for community when v_{max} added to $C^{t_i}_{current}$
$v_{max} \in \mathcal{U}^{t_i}$	Vertex maximizing $\mathcal{R}^{t_i}_C$
$\mathcal{R}_{max}[\dots]$	List containing all $\mathcal{R}^{v_i}_{max}$

Table 7.1: Notations used and their description

Algorithm 1 *iTop* – An algorithm to mine topic-centric interaction based communities on Twitter

Input : $\mathcal{T}$
Output : $C^{t_i}_{final}$, $\mathcal{G}^{t_i}_{visited}$
State : $C^{t_i}_{current}, \mathcal{U}^{t_i}, \mathcal{B}^{t_i}, \mathcal{G}^{t_i}_{current}, \mathcal{U}^{t_i}, \mathcal{R}_{v_i}, \mathcal{R}^{v_i}_{max}, \mathcal{R}_{max}[\ldots]$
Method :

for t_i in T **do**
  $C^{t_i}_{current}$ = search_bio(t_i)
  $\mathcal{U}^{t_i}, \mathcal{B}^{t_i}, \mathcal{G}^{t_i}_{current} \leftarrow$ expand($C^{t_i}_{current}$)
  while Continue_Condition($\mathcal{R}_{max}[\ldots]$) is *TRUE* **do**
    for v_i in $\mathcal{U}^{t_i}$ **do**
      $\mathcal{R}_{v_i} \leftarrow$ compute_localR($C^{t_i}_{current}, \mathcal{B}^{t_i}, \mathcal{G}^{t_i}_{visited}, v_i$)
    end for
    $\mathcal{R}^{v_i}_{max} \leftarrow$ get_maximum_R()
    $v_{max} \leftarrow$ get_maximum_RVertex($\mathcal{R}^{v_i}_{max}$)
    Update $\mathcal{G}^{t_i}_{current}, v_{max}$
    Update $C^{t_i}_{current}, v_{max}$
    Update $\mathcal{B}^{t_i}, v_{max}, \mathcal{G}^{t_i}_{current}$
    Update $\mathcal{U}^{t_i}, v_{max}, \mathcal{G}^{t_i}_{current}$
    add $\mathcal{R}^{v_i}_{max}$ to $\mathcal{R}_{max}[\ldots]$
  end while
  $\mathcal{G}^{t_i}_{visited} = \mathcal{G}^{t_i}_{current}$
  $C^{t_i}_{final} = C^{t_i}_{current}$
end for

Function 1 : To calculate continue condition

Continue_Condition($\mathcal{R}_{max}[\ldots]$):

$\epsilon \leftarrow \mathcal{R}_{max}[-1] - \mathcal{R}_{max}[-10]$
if $\epsilon \leq 0$ **then**
  return FALSE
else
  return TRUE
end if

Proposed Approach

Traditional approaches to detect communities of interest require the complete Twitter network $\mathcal{G}$ to be loaded into memory which may be infeasible in practice. Therefore, we investigate the use of the LM algorithm which makes use of local information to discover communities of interest [50]. Our proposed solution approach adapts the LM algorithm for directed graphs and consists of a four-block framework – *Warm Start, Expand, Filter, Iterate*. Our algorithm greedily maximizes an objective function called *local modularity* represented by $\mathcal{R}$ to find concentric communities around

an input topic. Algorithm 1 explains - *iTop* - our proposed solution approach to discover topic-centric communities on Twitter based on interactions between users. We now discuss the general four-block framework of our proposed approach.

1. *Warm Start* – Twitter users once registered are requested to enter publicly available profile information like Name, Location, URL and Bio. *Bio* is an internet acronym for the word 'biography' whose dictionary meaning is *an account of someone's life*. Hence, a Twitter user's *Bio* is a good indicator of his interest and inclination. Therefore, *iTop* takes a topic t_i, indicating the topic around our community of interest, as input and performs a search on the *Bio* of Twitter users. These list of users are a true positive entry point to provide a *warm start* to *iTop*. The use of true positive initial entry points for a *warm start* has been successfully used in previous approaches [29, 118]. Line 2 in Algorithm 1 corresponds to this step. We consider these true positive nodes to be part of our community of interest and we represent this by $C^{t_i}_{current}$.

2. *Expand* – Twitter allows registered users to subscribe to other users' tweets using the *Follow* feature. We manually observe that users on Twitter *follow'* many users without interacting with them. We argue that a network formed using the interactions of users like @-messages and RTs are a closer representation of actual relationships. Previous work also shows that the actual social network of users is 'hidden' amidst the links declared by users via the *Follow* feature [74]. In addition, the goal of this work is to find topic-centric communities to assist agencies like Digital Marketing and Law Enforcement. Such agencies would like to know the true social structures of the network. Therefore, in this step *iTop* expands the users taken as input from the previous step based on their interactions. *iTop* adds a directed weighted edge between two users only if there's an interaction between them via @-messages or RTs. For example, *iTop* adds a directed edge from user X to user Y with weight w only if X has interacted with Y where w represents the strength of the interaction. Due to the expansion, we now have $\mathcal{G}^{t_i}_{current}$ which consists of two node sets : $\mathcal{C}^{t_i}_{current}$ and $\mathcal{U}^{t_i}$ such that, $\mathcal{C}^{t_i}_{current} \cap \mathcal{U}^{t_i} = \phi$ and $\mathcal{C}^{t_i}_{current} \cup \mathcal{U}^{t_i} = \mathcal{V}^{t_i}_{current}$. Further, we also have $\mathcal{B}^{t_i} \in \mathcal{C}^{t_i}_{current}$ which contains nodes which are directly connected to at least one node in $\mathcal{U}^{t_i}$. Line 3 in Algorithm 1 corresponds to this step.

3. *Filter* – In this step, *iTop* iterates through each node $v_i \in \mathcal{U}^{t_i}$, taken as input from the previous step. For each v_i, *iTop* then computes the value of a pre-defined objective function called local modularity $\mathcal{R}$. Formally, $\mathcal{R} \propto \frac{1}{\mathcal{B}^{t_i}}$ and is a measure of the sharpness of $\mathcal{B}^{t_i}$ [50]. $\mathcal{R}_{v_i}$ calculates the ratio of weighted edges which have no nodes in $\mathcal{U}^{t_i}$ to that of weighted edges with at least one node in $\mathcal{B}^{t_i}$ for all v_i. The goal is to greedily maximize $\mathcal{R}^{t_i}_{\mathcal{C}}$; therefore $\mathcal{R}^{v_i}_{max}$ is added to $\mathcal{C}^{t_i}_{current}$. Lines 5-17 in Algorithm 1 corresponds to this step.

4. *Iterate* – We now repeat steps 2 and 3 until we don't reach the stopping condition. The stopping condition checks if $\mathcal{R}^{v_i}_{max}$ is stable or consistently negative. Line 4 in Algorithm 1 corresponds to this step.

The output of the four-step framework is the final desired community $C^{t_i}_{final}$ for the input topic t_i.

7.5 Experimental Dataset and Aids

InfoChimps – Dataset

Twitter provides a REST API service to download publicly available Twitter profiles and their interactions.But, Twitter imposes a strict 350 requests per hour account-based limit on authenticated API calls and 150 requests per hour IP-based limit on unauthenticated API calls. Due to these stringent API limits, we looked around for third party services which provide access to Twitter data at relatively lesser restrictions and affordable costs. *InfoChimps* is a BIG DATA access provider which allows customers to access their data via a *freemium* based subscription model. It provides access to both (free and paid) ready-to-download and API access-based datasets across multiple domains including social media. *InfoChimps* maintains a scrape of the Twitter network graph which is accessible via a RESTful web service. We access the *InfoChimps* dataset via RESTFul web services. The use of such a data does not hamper the generalizability of our experiments and we work on this snapshot for convenience.

FollowerWonk – Twitter Bio Search

The first step of our algorithm is to search the *bios* of Twitter users for each input topic t_i. *FollowerWonk* is an online Twitter analytics service which provides various services like social graph analysis and visualization, follower relationship performance and specialized Twitter searches on profile fields like URL, Bio and Location.[4] Therefore, the *search_bio function* (Line 2 in Algorithm 1) uses *FollowerWonk* to receive the initial community $C^{t_i}_{current}$ for the input topic t_i.

StrongLinks – Interaction Based Metric

StrongLinks is a metric provided by the InfoChimps API which measures the frequency of interaction between two users based on @-messages or RTs.[5] *StrongLinks* API takes as input a Twitter user and returns a list of up to 1000 users most interacted with. It also gives the total strength of the interaction which is directly proportional to the frequency of the interaction between the users. *StrongLinks* are directional and hence, may or may not be mutual between a pair of users. Therefore, we use the *StrongLinks* metric accessible via the REST API and form a directed edge from user X to user Y iff Y is a *StrongLink* of X. A *Strong Link* represents a combination of heterogenous interactions between two users X and Y. *StrongLinks* does not reveal the way they calculate the strength of interactions. However, the goal of our work is not to formulate a new interaction-based metric but rather to use existing metrics to discover topic-centric communities.

[4] `http://followerwonk.com/`

[5] `http://www.infochimps.com/datasets/twitter-graph-metrics-stronglinks`

7.6 Experimental Setup and Results

Setup

Topics $\mathcal{T}$ – CAD and Kashmir

We performed case studies on two diverse topics – *CAD* (Computer Aided Design) and *Kashmir* (a disputed region in the northern part of India). We chose these topics to suit two real-world use cases : Digital Marketing and Law Enforcement Agencies. For example, the digital marketing department of AutoCAD, a software application for CAD, would like to know the Twitter community around the topic *CAD*. Moreover, the topic *CAD* also helps us benchmark our results with a related previous study [81]. *Kashmir* is a disputed region in the northern part of India and has been rife with activism since the India-Pakistan partition in 1947. A law enforcement agency may want to monitor social media websites like Twitter to discover cyber-communities consisting of users talking about *Kashmir*.

StrongLinks

StrongLinks measures the degree of interaction between two users depending on their interaction. The InfoChimps *StrongLinks* API provides a maximum of 1000 links per user if available. But, due to limitation of economic resources (for API pricing), we perform experiments by expanding only top-k strong links in the second-step of the framework in our proposed approach – *Expand*. Previous research shows that a person can have roughly 150 stable social relationships in the real world (famously called the 'Dunbar Number') [107]. Studies also show that a person can have 6 close relations, 15 general weekly relationships and 50 monthly relationships. Recent work shows that these numbers also hold true for Twitter [66]. Therefore, we performed experiments on the values of k = 6, 15, 50 and 150 viz. expanded only the top-k *StrongLinks* in the *Expand* step. It must be noted that our framework is generalizable and one can choose to expand all available *StrongLinks*.

Results

We performed experiments on two topics - 'CAD' and 'Kashmir'. In case of the topic 'CAD', we used the same list of users for a *Warm Start* as given in [128]. We do this to compare our approach with the algorithm proposed by Tscherteu and Langreiter. For the topic 'Kashmir', we searched the bio of users using the *FollowerWonk* service and picked the top 10 users by virtue of their number of *StrongLinks*. We then performed the *Expand*, *Filter* and *Iterate* steps of our framework till the stopping condition was reached. Figure 7.1 and 7.2 show the varying local modularity values $\mathcal{R}$ versus the size of the community. For each run of k, the best $\mathcal{R}$ is computed and compared against other k-runs. A high $\mathcal{R}$ value (¿0.7) suggests high structural cohesiveness and hence, $\mathcal{R}$ with the maximum value is the best cohesive community around the topic [97]. For the topic 'CAD', the most cohesive community is formed at 57 nodes with $\mathcal{R}$ = 0.71 while for the topic 'Kashmir', the most cohesive community is formed at 133 nodes with $\mathcal{R}$=0.82. In both

versions, we found $\mathcal{R}$ to be poorly performing at k = 150. Overall, we observed that the structural quality (in terms of $\mathcal{R}$) of a community decreases as the community size increases. Leskovec et al. observe a similar pattern in a large scale evaluation of social networks [86]. Their results show that there's an inverse relationship between community quality and community size and nodes in large communities 'blend-in' to form a community.

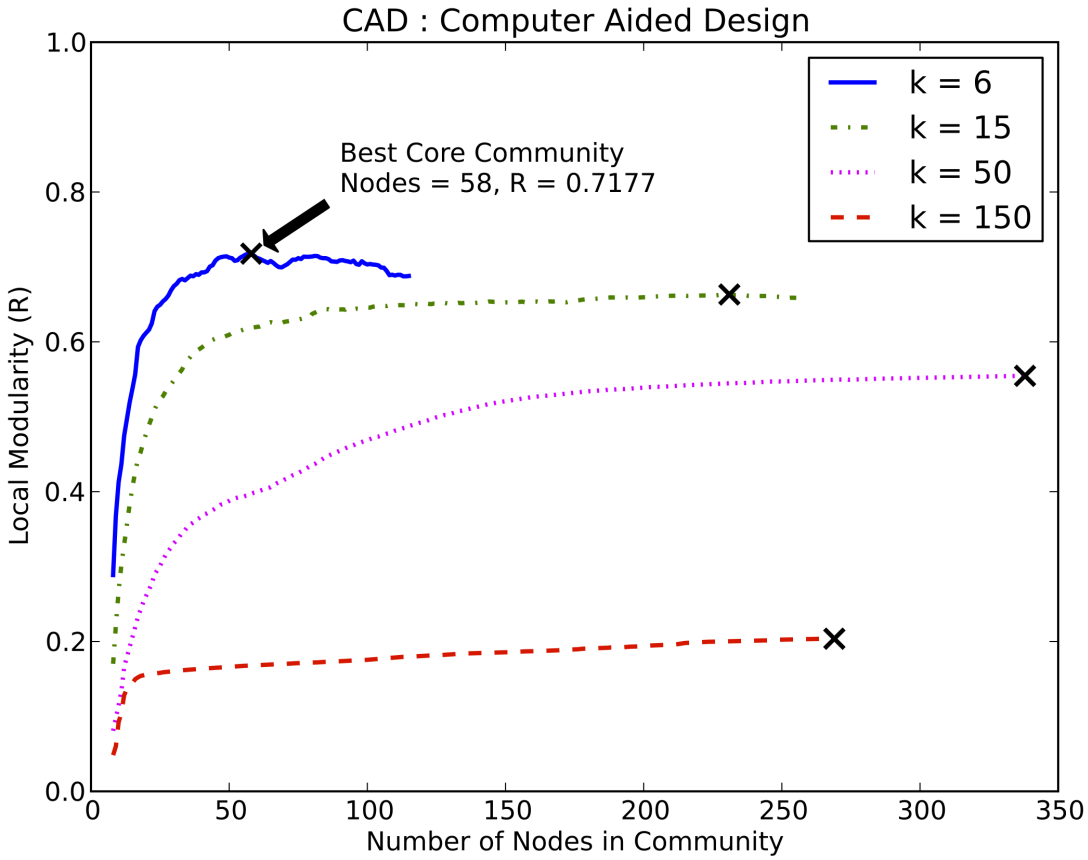


Figure 7.1: The graph shows the variation in local modularity $\mathcal{R}$ with the values of k = 6, 15, 50, 150 for the topic 'CAD'. The best structural community is formed at k=6 with nodes = 58 and $\mathcal{R}$=0.71.

Evaluation

Benchmark Comparison

We compare our proposed algorithm *iTop* to the network-degree statistic based heuristic algorithm (we would refer to it as CCD) proposed by Tscherteu and Langreiter [128]. In order to perform a fair comparison, we adapted the CCD algorithm to make use of the *StrongLinks* as edges rather than the follower graph. Therefore, the nodes are represented as Twitter users and the directed

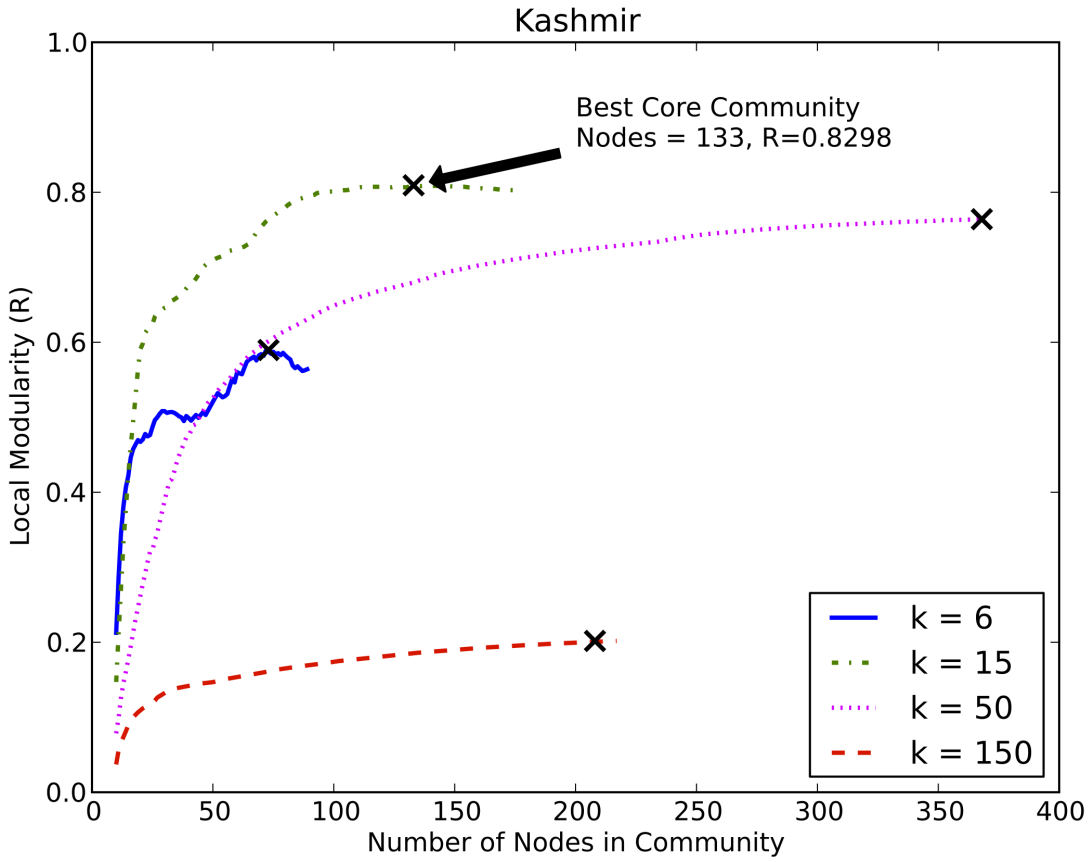


Figure 7.2: The graph shows the variation in local modularity $\mathcal{R}$ with values of k = 6, 15, 50, 150 for the topic 'Kashmir'.The best structural community is formed at k=15 with nodes = 133 and $\mathcal{R}$=0.8298.

weighted edges are based on interactions. We compare the best versions of *iTop* viz. k=6 for 'CAD' and k=15 for 'Kashmir' to the CCD algorithm. We provide both algorithms with the same *Warm Start* viz. set of seed users. We use the same stopping condition for both *iTop* and *CCD*. Similar to the earlier versions, we compare the values of local modularity $\mathcal{R}$ to decide the winner. Figure 7.3 shows the comparison of the *iTop* versus *CCD* algorithm with respect to the $\mathcal{R}$ value.

On both topics, the $\mathcal{R}$ values obtained using *iTop* (k=6 : 0.7177 for 'CAD' , k=15 : 0.8298 for 'Kashmir') are better than *CCD* (k=6 : 0.5211 for 'CAD', k=15 : 0.3652 for 'Kashmir') . The results show that the *CCD* algorithm (for both topics) never improves the structural quality of the community beyond the first step. We argue that the reliance of the CCD algorithm on the follower graph of the network rather than the interaction graph leads to formation of an in-cohesive structure. It also confirms that interactions on Twitter based on @-messages, RTs in Twitter reveal the true underlying social structure.

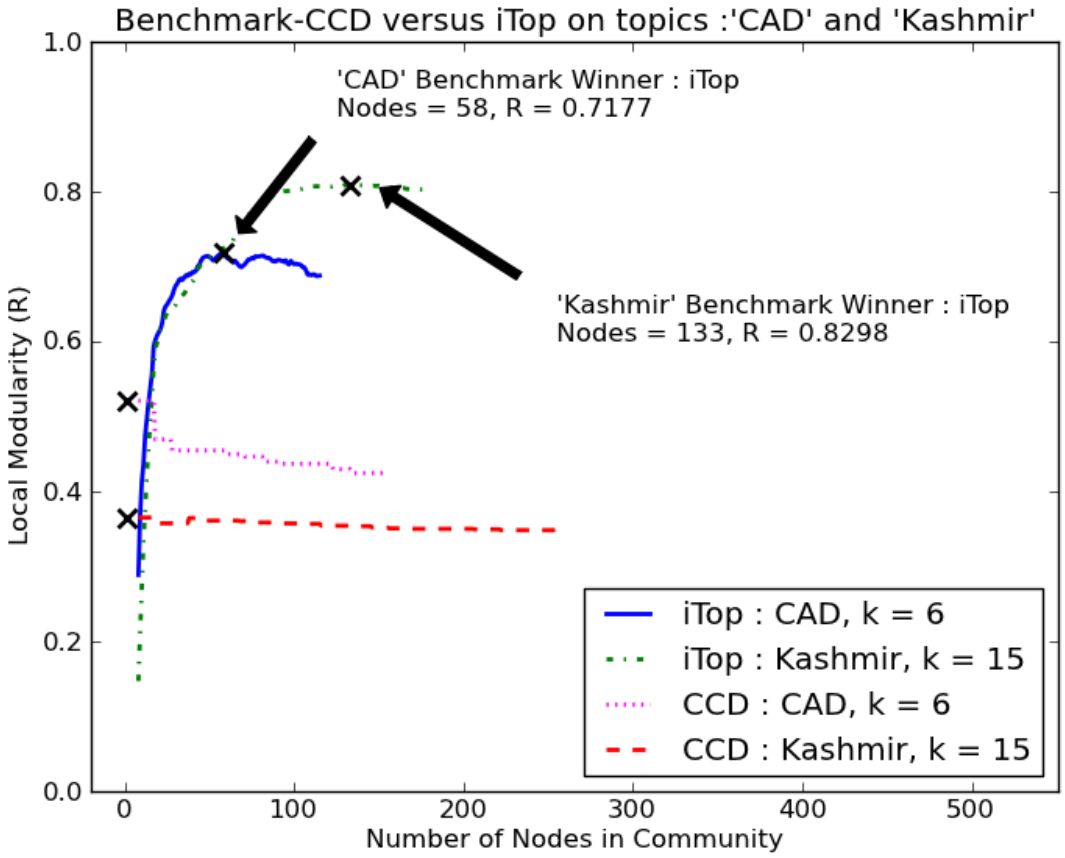


Figure 7.3: The graph shows the comparison of the variation in local modularity $\mathcal{R}$ with values of k= 6 for the topic 'CAD' and k=15 for the topic 'Kashmir' using the algorithms *iTop* and *CCD*.

Topic Based Semantic Evaluation

In order to evaluate if the discovered communities are central to the input-topic we perform a semantic evaluation. We check the frequently used n-grams in tweets and manually inspect their semantic relation to the input topic. Infochimps Twitter Wordbag API[6] provides a web service which returns a list of frequently used terms by the user. We retrieve these frequently used terms for the users in the output community and draw a tag cloud. Figure 7.4 and 7.5 show the tag clouds for the communities on topics 'CAD' (k=6) and 'Kashmir' (k=15) respectively.

We manually inspect the frequent n-grams in the users tweets on the 'CAD' Wiki page to check the semantic relation between terms. For example, in Figure 7.4 the word 'plm' is an acronym for Product Lifecycle Management, a process in which CAD tools are extensively used. The word 'simulation' occurs in the tag cloud as CAD is also used to simulate, design and engineer before a product is released. 'SolidWorks' is a firm which provides a 3D CAD software to design graphical

[6] `http://www.infochimps.com/datasets/twitter-wordbag`

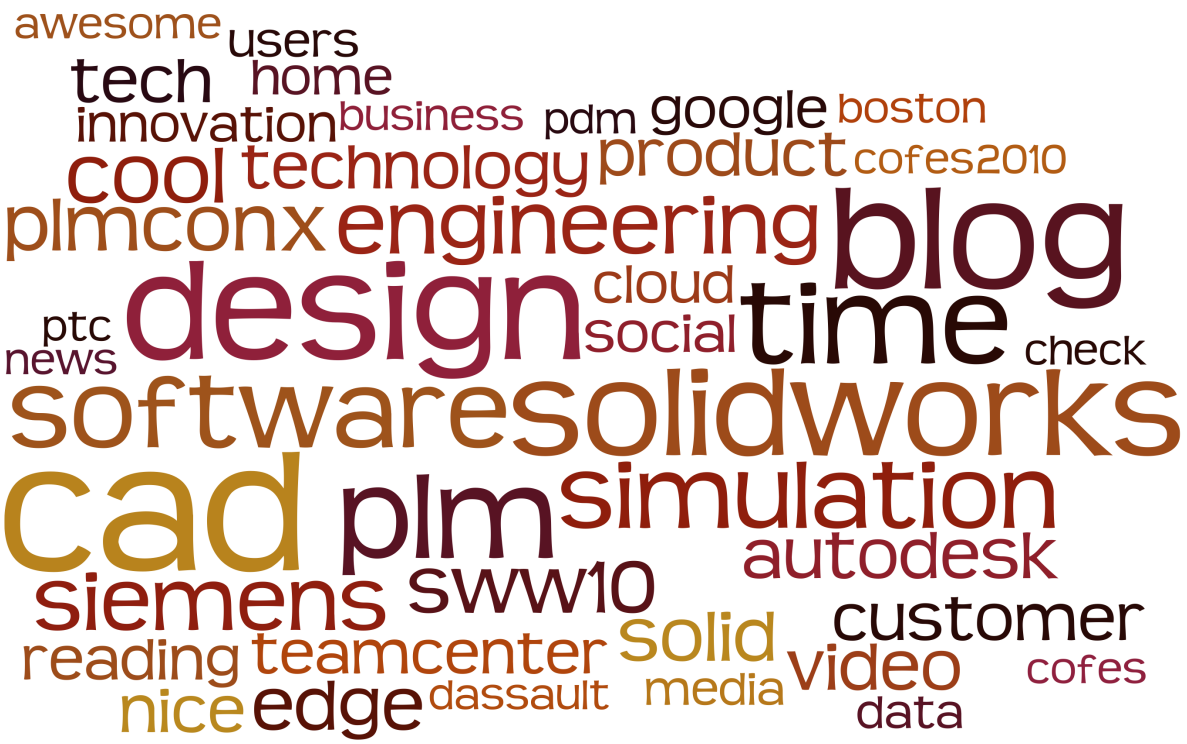


Figure 7.4: Tag cloud of the frequently used n-grams for the Twitter users in the community with topic 'CAD' (k=6)

prototypes. 'Siemens' is a German company which sells PLM software. We perform a similar manual inspection of the frequently used n-grams in the 'Kashmir' community. For example, the words 'pakistan', 'lahore' and 'karachi' are found in 7.5 as 'Kashmir' is a disputed region between India and Pakistan. 'Kashmir' is also a central political topic in Pakistan and hence the presence of words like 'news' and 'zardari' (Asif Ali Zardari the current president of Pakistan[7]). Therefore, manual evaluations of both communities reveal that frequently used n-grams of users are semantically related to the input topic.

Empirical Evaluation

We now evaluate the efficiency of *iTop* based on user profile information on Twitter. We requested three independent annotators to manually go through each member in the community and check their profile information like bio, lists and tweets. We then ask them to perform three checks based on the users tweets – (1) Presence of input topic in Bio, (2) Relevance of user to the input topic based on tweets (3) Membership or subscription to topic-relevant lists. In each of these three cases, we ask our annotators to mark either : 'relevant' (1) or 'not relevant' (0). In case of conflicts, we arranged a meeting with all the three annotators and discussed the issues ironing our

[7] http://en.wikipedia.org/wiki/Asif_Ali_Zardari

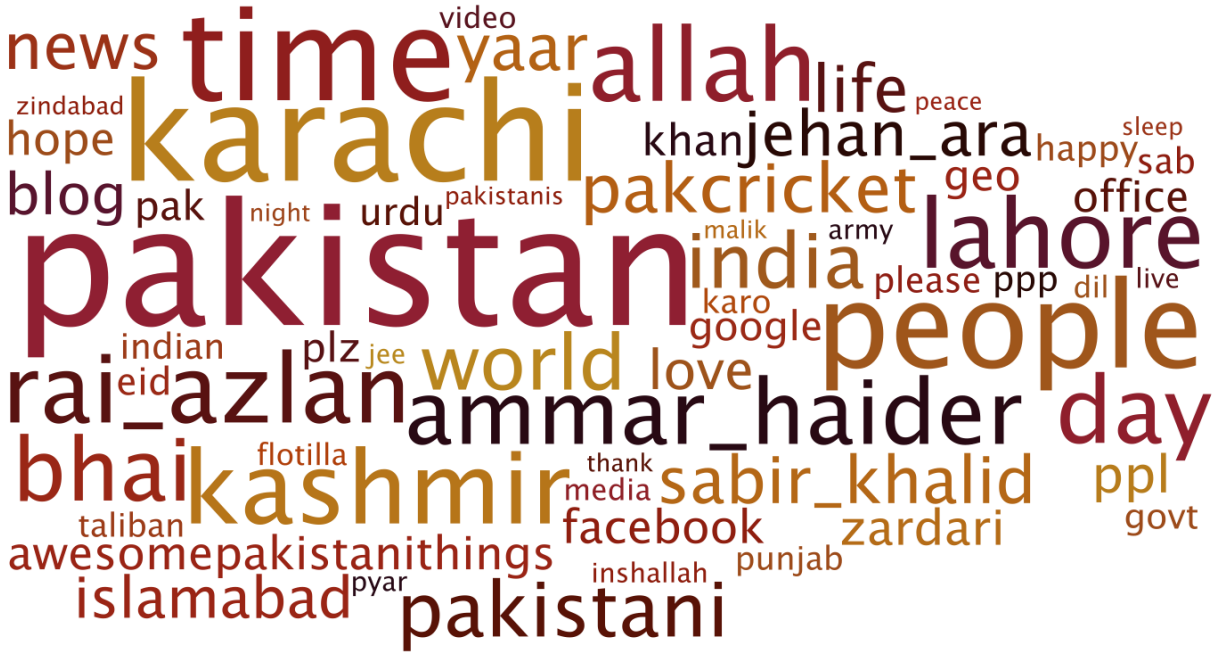


Figure 7.5: Tag cloud of the frequently used n-grams for the Twitter users in the community with topic 'Kashmir' (k=15)

disagreements in the process. Due to the arduous nature of this task, we perform this evaluation for only the communities with best structural cohesiveness on the topics : 'CAD' : k=6 and 'Kashmir' : k=15. Table 7.2 shows the summary of the manual evaluation. The seed nodes were excluded in this evaluation.

	Percentage of users	
	CAD (k=6)	**Kashmir** (k=15)
Bio contains input topic	17%	0.8%
Relevant to input topic	84%	81%
Members of a topic-relevant list	80%	70%

Table 7.2: The table summarizes the manual evaluation of users across multiple dimensions in the 'CAD' (k=6) and 'Kashmir' (k=15) communities.

For both input topics 'CAD' and 'Kashmir', we observe a very small percentage of users (CAD : 17% , Kashmir : 0.8 %) which have the input topic in their bio. But, the content posted by these users show that they are relevant to the input topic. 'CAD' contains 84% 'CAD'-relevant users in the community while 'Kashmir' contains 81% 'Kashmir'-related users. Moreover, 80% of users in the 'CAD' community are members or are subscribed to a 'CAD'-relevant list while the 'Kashmir' community contains 70% of users in a 'Kashmir'-relevant list. This shows that *iTop* is efficiently

able to discover topic-centric communities even if the user hasn't explicitly mentioned his interests in his profile-bio.

Social Network Analysis

Clustering Coefficient, Average Path Length and Graph Density

Figure 7.6 and 7.7 shows the network graph of the topic-centric communities for 'CAD' and 'Kashmir' respectively. We also calculate network statistics for both the communities. Table 7.3 contains the values of such statistics like Network Diameter, Average Clustering Co-efficient and Graph Density. We see that the average clustering co-efficient in both communities is high (normal networks $\sim$ 0.3) indicating a strongly connected structure amongst the group. A low average path length ($\sim$2) also shows that a node in the community can reach any other node in a maximum of 2 hops thus indicating a highly cohesive group. Both networks also display a relatively high graph density.

	CAD (k=6)	**Kashmir** (k=15)
Nodes	57	132
Edges	1292	4444
Average Degree	22.67	33.67
Network Diameter	5	7
Graph Density	0.405	0.257
Average Clustering Coefficient	0.712	0.54
Average Path Length	1.70	1.94

Table 7.3: The table summarizes network statistics for the topic-centric communities –'CAD' (k=6) and 'Kashmir' (k=15).

Betweenness, Degree Centrality

Users who have *Betweenness Centrality* act as brokers or information connectors in the community. Users with high *Degree Centrality* indicate high connectivity to the community while users with high betweenness centrality represent single point of failure. Figure 7.8 shows the scatterplot distribution of betweenness versus degree centrality for the topic-centric community 'Kashmir' (k=15). The scatter plot illustrates that users with high degree centrality need not be the nodes with high betweenness centrality. Such insights can be useful to security analysts in law enforcement agencies. For example, if a security analyst would like to cripple information flow in a hate-speech network he should target users with high betweenness centrality rather than the users with high degree centrality.

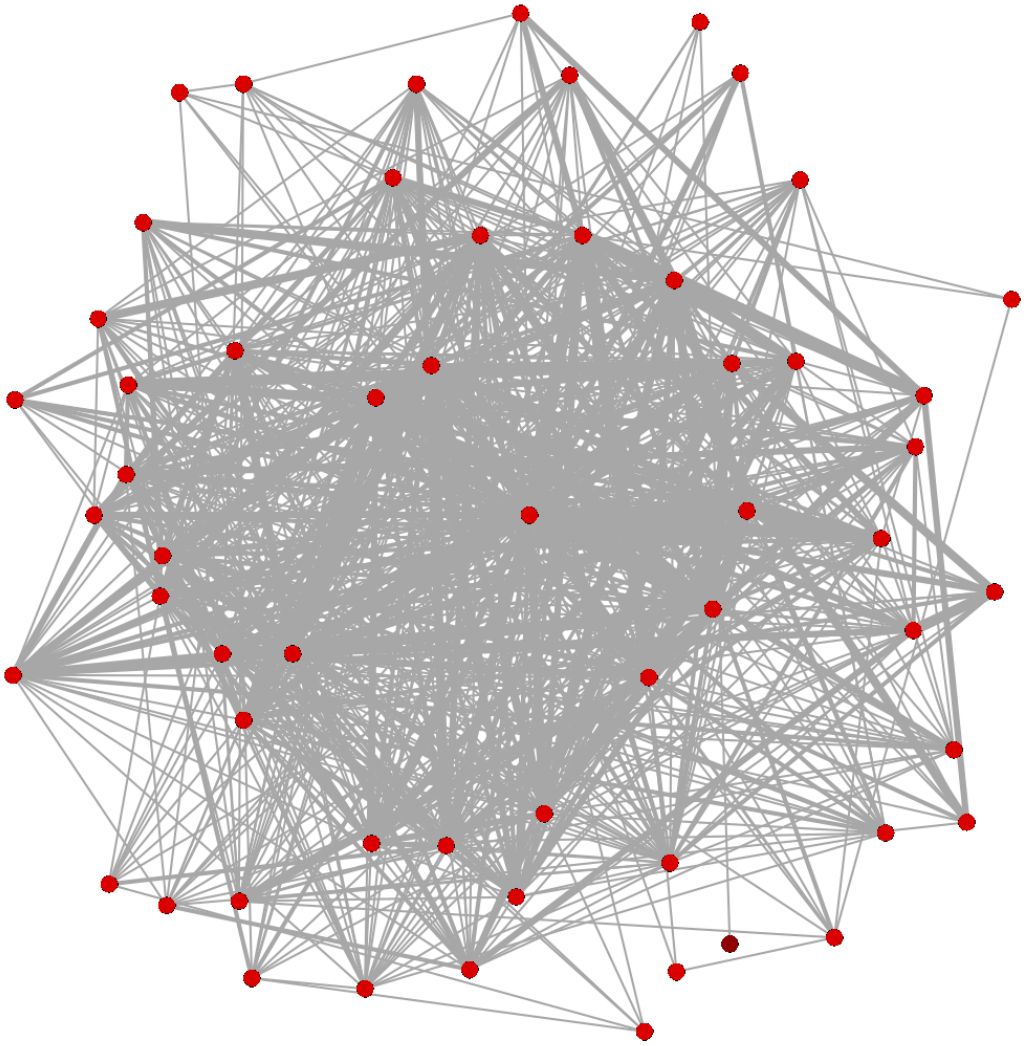


Figure 7.6: The figure shows the network graph of the topic-centric community 'CAD' for k = 6.

Authoritative, Hub Centrality

Figure 7.9 and 7.10 shows the top-5 users measured on the basis of *authoritative centrality* and *hub centrality* in the 'CAD' (k=6) community. Authorities are users with a high number of incident links while hubs point to authorities. Authorities and hubs are mutually co-dependent on each other. Authorities and hubs can give non-trivial community insights to digital marketing analysts to answer questions like : "Which users lead the community of my product?", "Who forms the high tier leaderships in my community?", "Who are the most influential users using my product?".

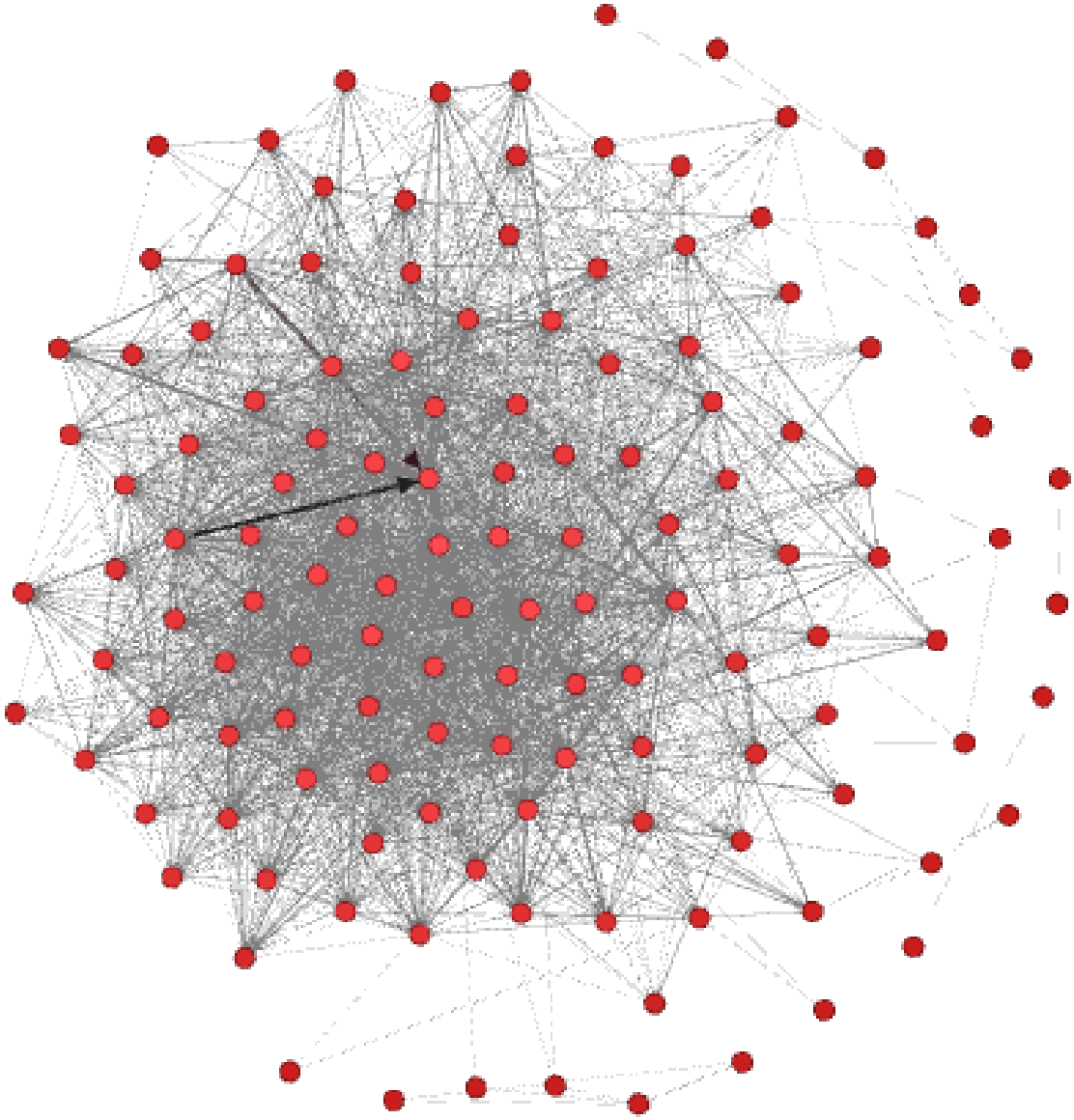


Figure 7.7: The figure shows the network graph of the topic-centric community 'Kashmir' for k = 15.

Core – Periphery Pattern

Core-periphery patterns widely exist in economic markets, organizational structures and scientific networks [42]. A core periphery consists of two distinct set of user classes : core and periphery. The core users are important users in the network while the users on the periphery are relatively less influential. Figure 7.11 shows the core-periphery pattern for the users in the topic community 'CAD' based on their betweenness centrality scores. This shows that only a few users in the

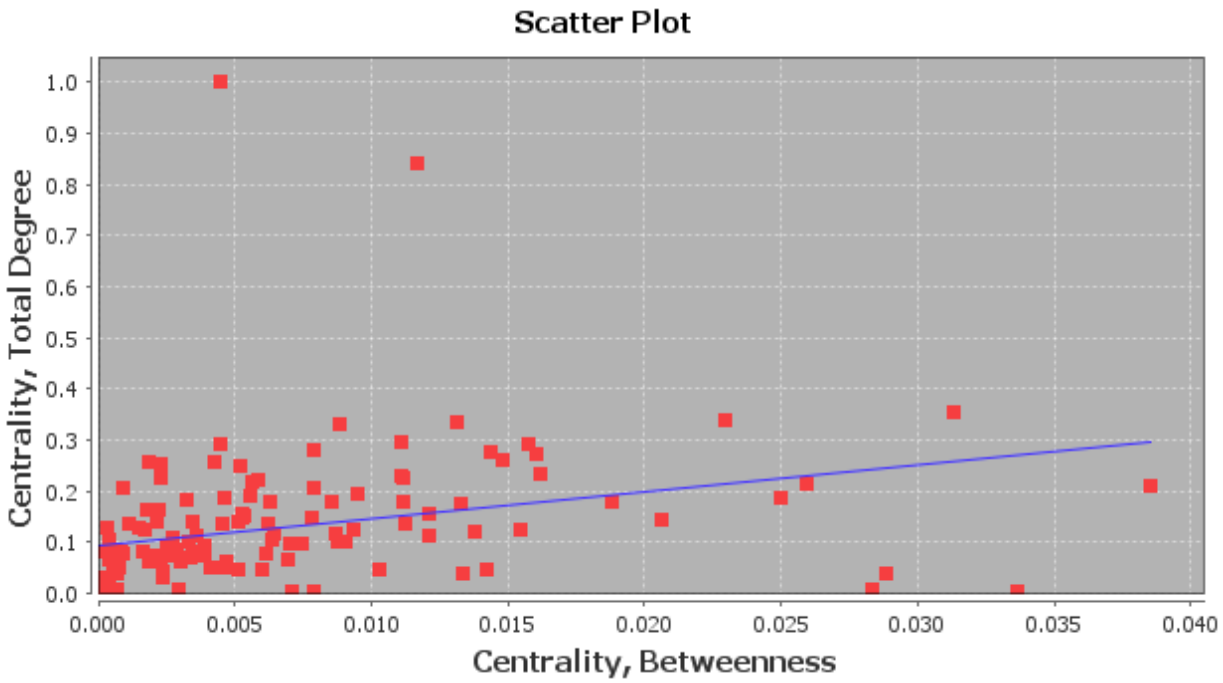


Figure 7.8: The figure shows the scatter plot for betweenness centrality versus degree centrality for the topic-centric community 'Kashmir' (k = 15).

Figure 7.9: The figure shows the top 5 nodes based on their authority centrality for the topic-centric community 'CAD' (k = 6).

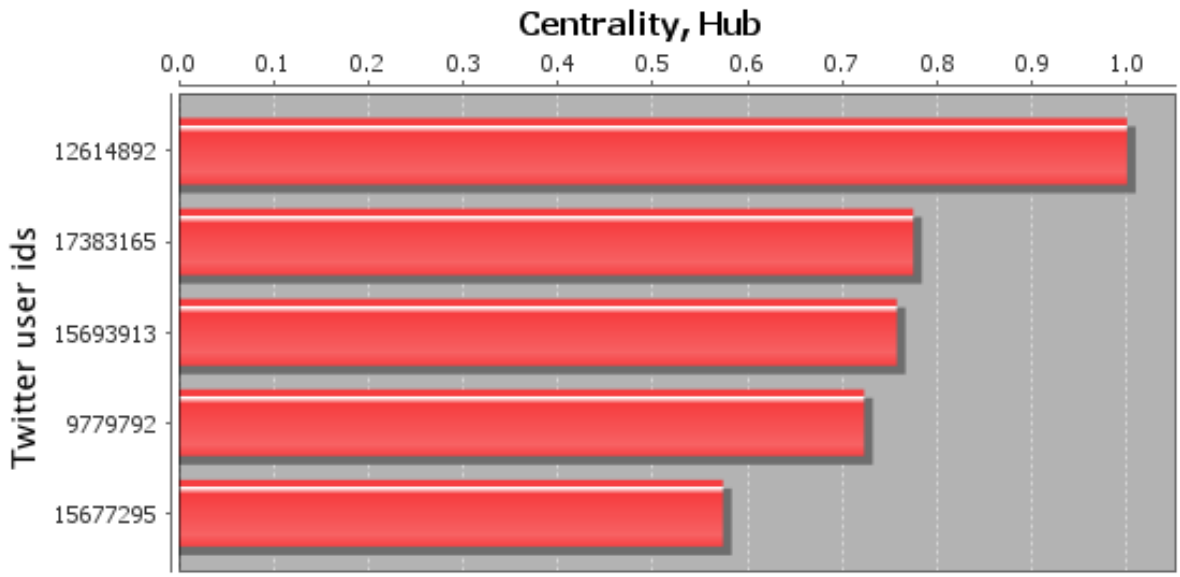


Figure 7.10: The figure shows the top 5 nodes based on their hub centrality for the topic-centric community 'CAD' (k = 6).

'CAD' community hold important positions within the community while a majority of them have lesser influence. For example, the core users could be targeted by digital market analysts to spread information about their products and services to the wider community.

7.7 Future Work

Specific *Big Data* application scenarios like digital marketing and law enforcement agencies would like to discover or extract user communities around relevant topics. However, they would also like to know the temporal behavior of these communities. For example, a security analyst or a digital marketing agency would like to find out answers to questions like - how a community was evolved over time and intra-community interaction? We plan to study this phenomena, understand if there's a particular interaction pattern and develop a model to capture it. In addition, we would also like to understand the behavior of influential users as well as users who are on the borderline or periphery of such communities. If such periphery users are detected early, we could develop different ways to influence their decision in or against the favor of joining the community depending on the application scenario.

7.8 Conclusion

We present a novel approach *iTop* to discover topic-centric interaction based communities on Twitter. We experiment on a dataset provided by InfoChimps via a web service and perform a case study

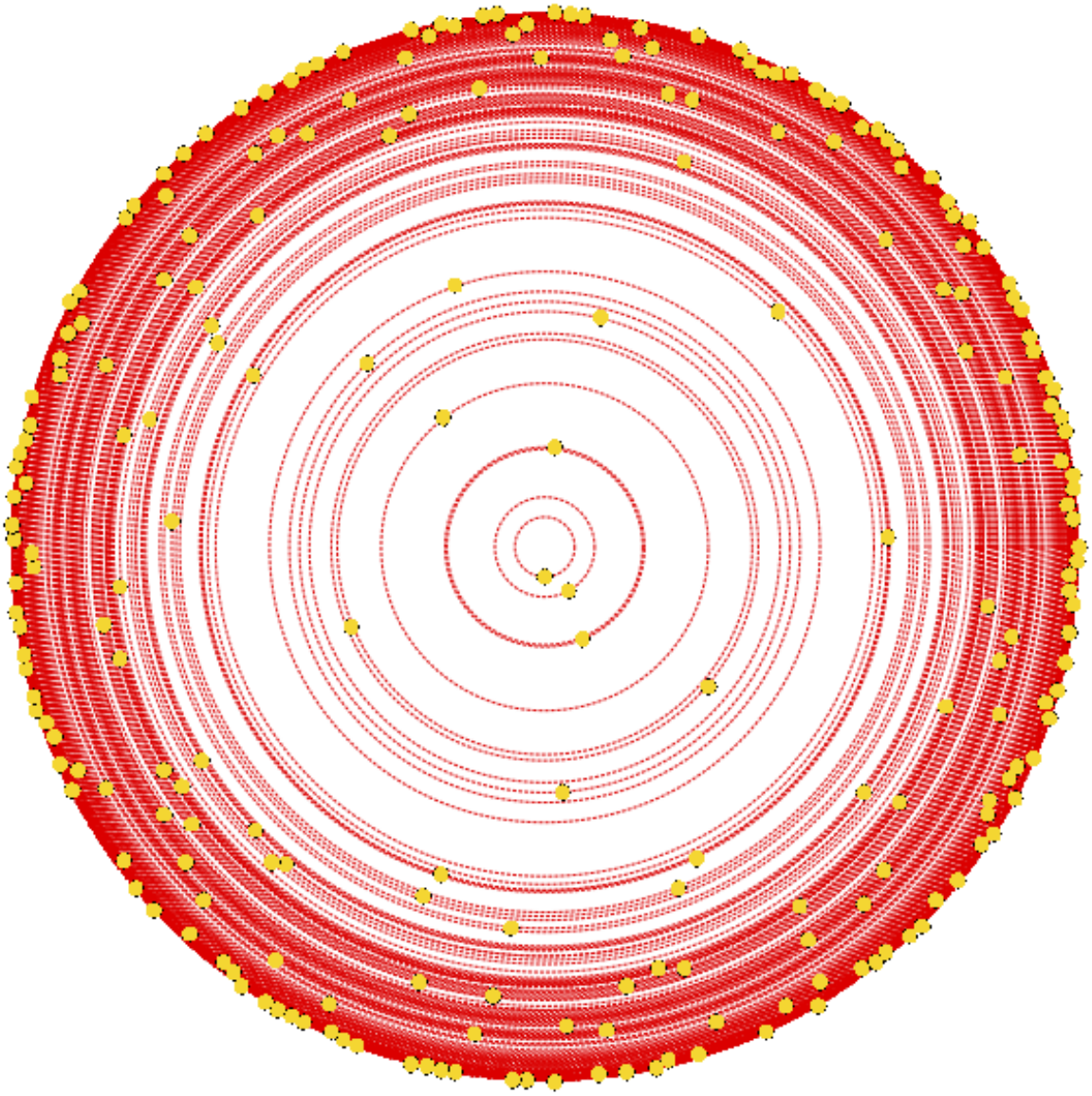


Figure 7.11: The figure shows the core-periphery layout for the users in the topic community 'CAD' based on betweenness centrality.

on two diverse topics. We evaluate the output of *iTop* across three dimensions – structural, topic-semantics and empirical. Results show that *iTop* can be efficiently used to discover topic-centric interaction based communities.

Chapter 8

Tag Recommendation on Social Media

In continuation of the previous chapter for Information Retrieval enhancement, we further look at a complementary approach to enhance multimedia information retrieval. Concretely, we look at recommendation of tags (or keywords) on social multimedia websites. We investigate the use of Twitter to enhance *tag* recommendation capabilities on social media websites.

8.1 Research Motivation and Aim

Contemporary Web 2.0 websites like social media thrive on user generated content. Each piece of content is accompanied by various metadata information like time and location. Several social media websites like Flickr, Youtube and SoundCloud allow its users to add *tags* (also called as labels, keywords) to describe the content (photo, video, songs etc.). *Tags* provide contextual information and meta data to the content. Therefore, the presence of appropriate tags could facilitate search, information retrieval, object classification, content discovery and exploration [39, 78, 80, 95, 126]. However, prior studies show that user supplied tags are insufficient and low quality in nature [80, 89, 95, 60].

Garg *et al.* report that most of the time users add very few tags or even none at all. Their study on Flickr photo sharing system indicates that at least 20% of public photos have no tag at all [60]. Sigurbjornsson and van Zwol report that 64% of the photos have 1-3 tags (a study consisting of over 52 million publicly available Flickr photos) [114]. Song *et al.* perform a study on tagging on CiteULike and report that the average number of tags per paper was 3.35 (a study consisting of 32242 entries, with 9623 distinct papers) [115]. Zhuang *et al.* reveal that web images are usually not annotated with proper tags. Several images are completely unlabeled and tags descriptions are often incomplete [139]. Some previous studies also show that several tags in Flickr are imprecise and around 50% of the tags are unrelated to the image content [80, 89]. Moxley *et al.* reveal that user-generated annotations on web video repositories such as YouTube are not quality-controlled (annotations are typically incomplete and noisy, incorrect keywords, missing quite a few relevant ones) [95]. Toderici *et al.* present a system that automatically recommends tags for YouTube videos whose titles are too short or whose descriptions and tags are either brief or missing (calling

the phenomenon as *metadata deserts*) [126].

These research findings clearly indicate major quality issues with free-form user-provided web-resource tags on various social media Web 2.0 platforms. The quality of user-supplied web-resource tags (precision, completeness, quantity, relevance, richness) have a profound impact on the efficacy of web-search, navigation, browsing and recommendation systems exploiting these tags for decision making. In order to address the *tag quality* problem in online social media services several techniques have been proposed for *automatic tag recommendation and annotation* (the focus of this work) for user-generated web-resources (images, weblogs, songs and videos). In this work, we propose a *Wisdom of Crowd* mashup solution framework for tag recommendation on online social media services. We investigate the use of Twitter to recommend tags for Flickr[1], Photobucket[2] (image hosting and sharing), YouTube[3], Dailymotion[4] (video hosting and sharing) and SoundCloud[5](audio hosting and sharing). The research motivation of this work is to investigate alternate and effective automated solutions to semantically tag a phenomenal amount of inadequately annotated user-generated web-resources and address the *tag quality* problem. The *specific research aim* of this research is to investigate the application of mining tweets to semantically enrich the *tags* of popular image(Flickr, Photobucket), video(YouTube, Dailymotion) and songs(SoundCloud) sharing services.

8.2 Research Questions

Twitter (defined as real-time information network[6]) is a very popular and fast growing micro-blogging service which allows users to post stream of messages of up-to 140 characters in length (called *tweets*) and follow other users (called *Twitterers*). Twitter has attracted worldwide popularity. As of June 2011, on an average there are 200 million tweets posted per day[7] and an average 460,000 new accounts were created per day during the month of February 2011[8]. *Twitterers* also use various external social media services like Flickr and Photobucket, Youtube and Dailymotion, SoundCloud in order to share photographs, videos and songs.

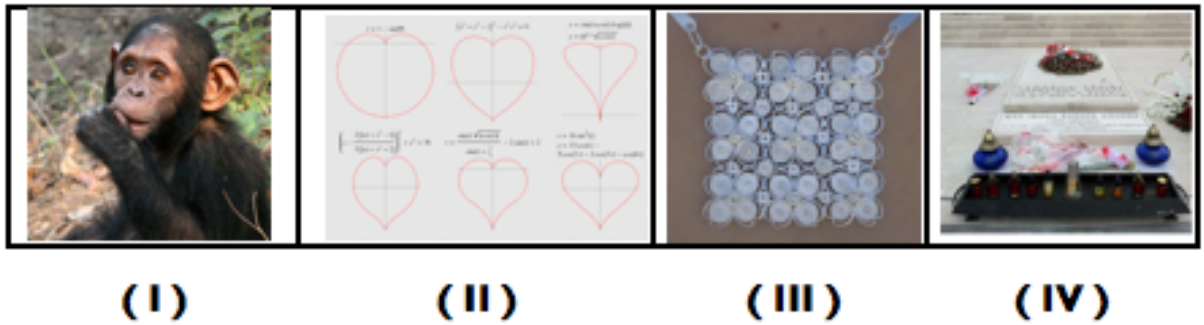
Twitter defines hashtags as keywords or topic marked by a user in a tweet [9]. Hashtags are preceded by the # symbol and are used to categorize the tweet. Hashtags can occur anywhere within the message body and this feature enables convenient searching as users can click on a hashtag in the tweet to retrieve other tweets in that category. We manually inspected tweets containing URLs to external social media services (like Flickr, YouTube etc.). We notice that both hashtags and the tweet content can be used to label the associated multimedia object. Table 8.1 shows illustrative

[1] http://www.flickr.com/
[2] http://photobucket.com/
[3] http://youtube.com/
[4] http://www.dailymotion.com/
[5] http://soundcloud.com/
[6] http://twitter.com/about/
[7] http://blog.twitter.com/2011/06/200-million-tweets-per-day.html
[8] http://blog.twitter.com/2011/03/numbers.html
[9] http://support.twitter.com/entries/49309-what-are-hashtags-symbols

Table 8.1: Illustrative examples provided as evidences to show that the hashtags and content in tweets have potential to semantically enhance multimedia retrieval

Tweet	Hash Tags	Service Media	Media Tags
Chimp Photo of the Week: http://flic.kr/p/ab9NQy #chimp	**#chimp**	Flickr	JGI, Jane Goodall, Jane Goodall Institute, **chimp**, chimpanzee, Tchimpounga, sanctuary
Lovely **Heart** Curves [#VisualArt #Graphics #Illustration #Math #Curves #Shape] http://pbckt.com/p5.UEB3Av	**#VisualArt #Graphics #Illustration** #Math #Curves #Shape	Photobucket	**visual art**, **math curves**, **graphics**, **illustration**
Stunning and delicate necklace in #silver sterling #filigree http://pbckt.com/pL.dFmmux	**#silver #filigree**	Photobucket	No Tags
The **grave** of **General Anders** at the **Polish War Cemetery**, **Monte Cassino** #ww2 http://flic.kr/p/9LgBSq	**#ww2**	Flickr	No Tags

Figure 8.1: Snapshot of photos associated with tweets to illustrate motivating examples in Table 8.1



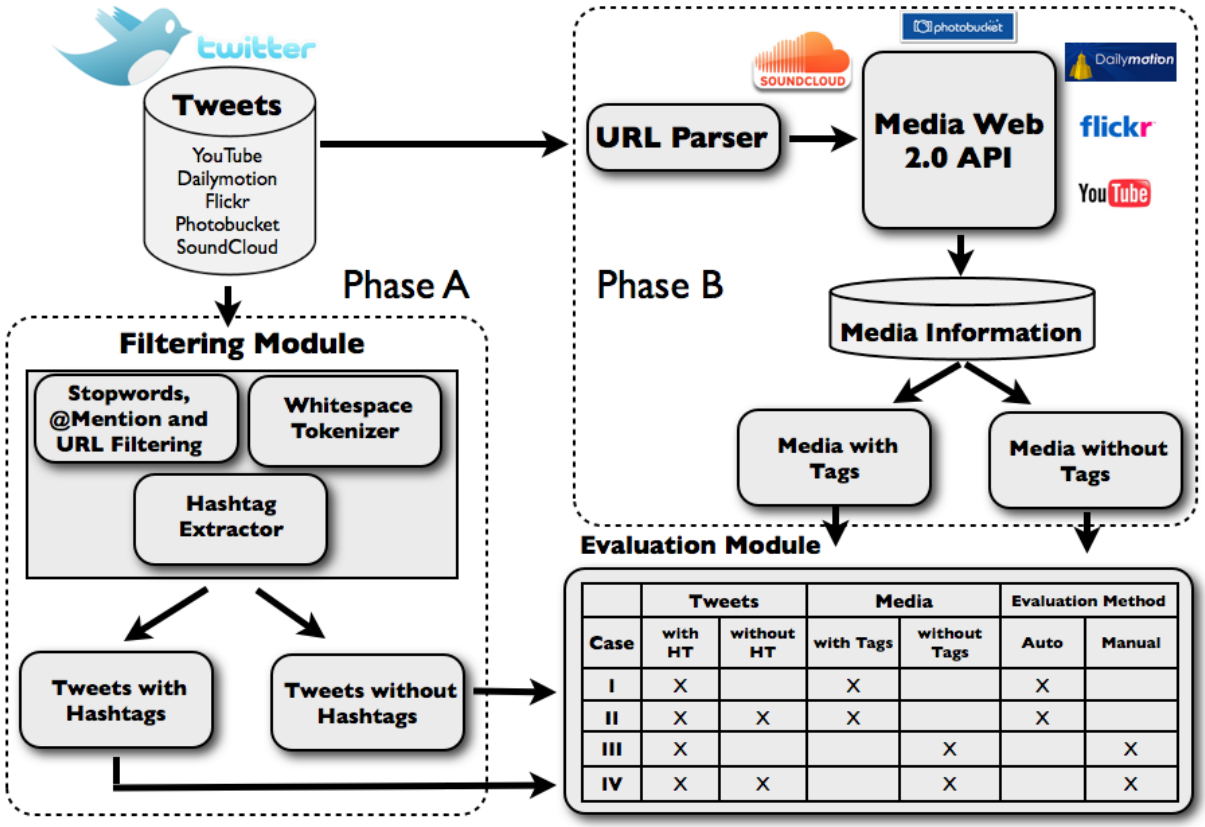
examples to explain our intuition. Figure 8.1 shows snapshots of photographs to examples in Table 8.1. Examples I and III show how hashtags in tweets can be used for tag enrichment while Examples II and IV demonstrate that content in tweets can be used for tag recommendation in the absence of tags in the associated multimedia objects. Based on our intuition we formulate and address the following research questions :

- **Research Question 1(RQ1):** Can *hashtags* in tweets be exploited to automatically tag external media objects in Flickr, YouTube etc.?
- **Research Question 2(RQ2):** Can *hashtags with content* in tweets be exploited to automatically tag external media objects in Flickr, YouTube etc.?

8.3 Methodology

In this section, we describe our solution approach and data collection.

Figure 8.2: Proposed solution framework for automatic tag recommendation on Flickr, Photobucket, YouTube, Dailymotion, SoundCloud by exploiting tweets



Solution Approach

Figure 8.2 details our proposed solution framework. The solution approach primarily consists of two concurrent phases – **Phase A** and **Phase B**. The output from both these phases are then passed to the **Evaluation Module** to obtain the results.

Phase A

The tweets from the datastore are passed to the *Filtering Module*. This modules first tokenizes each tweet using a *WhiteSpaceTokenizer*[10]. Next, all tokens containing URLs, stopwords and @-mentions(which indicate replies to a *Twitterer*) are filtered out. The above mentioned tokens wouldn't enhance or enrich multimedia and hence, we filter out such low quality content. We then segregate tokens from tweets which contain hashtag(s) from the ones without hashtag(s). We call these tokens **TTH** and **TTNH** respectively.

[10] `http://nltk.googlecode.com/svn/trunk/doc/api/nltk.tokenize.regexp.WhitespaceTokenizer-class.html`

Phase B

The tweets from the datastore are passed to the *URL Parser*. The parser filters out the URL and passes the URL to the *Media Web 2.0 API* module. This module calls the respective Web 2.0 API (Flickr [11], Photobucket [12], YouTube [13], Dailymotion [14], SoundCloud [15]) and collects the information from the media URL. We then segregate media objects which contain tag(s) from ones without tag(s). We call these **MT** and **MNT** respectively.

Evaluation Module

This module takes the input TTH, TTNH and MT, MNT and outputs scores on various evaluation metrics. Section 8.4 details the evaluation methodology and metrics used.

Data Collection

We use the *Twitter Search API*[16] to collect relevant tweets for our experiments. Table 8.2 shows the various terms used to query the *Twitter Search API*. We tried different variations of queries and selected the ones which gave the most unadulterated tweets in the results. For example, tweets starting with RT signify *Retweet* (similar to forward in e-mail). These can essentially be treated as duplicates and may not semantically enhance the associated multimedia object. Hence, we filter such tweets from our results. Due to limitations of the *Twitter Search API* we are able to obtain 7500 tweets for each social media service. We filter out non-English tweets by using the *iso_language_code* parameter option in tweets as returned by the API. Our approach is language independent, however, in this work we only perform experiments on English tweets. We then filter out duplicate tweets by performing a simple string match. The number of tweets for each media service is given in Table 8.3. For each social media service, we can divide the total number of tweets into four categories based on the presence or absence of hashtags in tweet and presence or absence of tags in the associated media object. Table 8.4 details the number of tweets present in various categories.

8.4 Evaluation

This section discusses the validation technique and evaluation metrics used in our experiments.

[11] http://www.flickr.com/services/api/
[12] http://photobucket.com/developer
[13] http://code.google.com/apis/youtube/1.0/developers_guide_python.html
[14] http://www.dailymotion.com/doc/api/rest-api.html
[15] http://developers.soundcloud.com/
[16] https://dev.twitter.com/docs/using-search

Table 8.2: Search terms used to query Twitter for different social media services

Service	Search Terms
Flickr	flic.kr -RT
YouTube	youtu.be -RT -uploaded -liked -favorited
Photobucket	pbckt.com -RT
DailyMotion	dai.ly -RT
SoundCloud	snd.sc -RT

Table 8.3: Dataset size for different social media services

Service	Number of Tweets
Flickr	2833
YouTube	2280
Photobucket	1487
DailyMotion	290
SoundCloud	5796

Table 8.4: Tweets belonging to different buckets depending on the presence or absence of hashtag and presence or absence of tags associated with media objects

			Tweets	
Media Type	**Service Media**		**With Hashtags**	**Without Hashtags**
Video	YouTube	with Tags	656	1566
		without Tags	14	44
	Dailymotion	with Tags	65	196
		without Tags	5	24
Image	Flickr	with Tags	613	1470
		without Tags	160	590
	Photobucket	with Tags	24	569
		without Tags	92	802
Music	SoundCloud	with Tags	189	2944
		without Tags	118	2545

Validation Technique

In order to perform validation on our dataset of tweets, as described in Table 8.4, we see two possible cases. Tweets : (i) with tags in associated media objects and (ii) without tags in associated media objects. For (i), we use an *automated* technique while for (ii) we use a *manual* technique. We now explain these techniques and motivation behind using them.

Automated Technique

One of the methods to measure the efficacy of an automated tag recommendation system is to compare the output of an automated solution against dataset which is pre-annotated or pre-tagged (a test dataset containing the ground-truth, web-resources tagged by the owner or the user who uploaded the object on the social media). Song *et al.* use evaluation metrics such as top-k accuracy, exact-k accuracy, tag-recall and tag-precision and assess the performance of their technique based on presence of tags recommended by their algorithm in the tag-set annotated by the user [115]. There are some limitations of this validation approach as a tag recommended by an automated solution not being present in the pre-annotated dataset does not mean that the tag is not relevant. Previous studies report that the average number of tags on web-resources is low (between 0-4) [60][114][115]. This naturally decreases the probability of a match between the predicted output and actual output for such resources. Also, due to presence of synonymy and morphological variations in natural language, a concept match may not mean a match in terms. However, we believe that the percentage of predicted tags by our technique being present in the pre-annotated tag-set (by the web-user) is a reasonable indicator on the quality of the automatic tag recommendation technique.

In this technique, we perform a simple string match by tokens obtained from tweets with tags obtained from the associated media (MT). We observe that a lot of tweets in our dataset are auto-generated due to user accounts from external social media services being *linked* (sharing of Activity Feed) to Twitter. Such tweets generate a common hashtag like #Snapbucket for Photobucket and #SoundCloud for SoundCloud. We do not consider such hashtag(s) from the tweets while performing the validation.

Manual Technique

As discussed by Garg *et al.*, one of the ways of evaluating the efficacy of a tag recommendation system is manual validation by users or human annotators [60]. While manual validation of system output by users has been a scientifically acceptable form of performance evaluation methodology, it is not perfect due to inherent subjectivity, missing context, human errors, different guidelines, different senses, disagreements between annotators and ambiguities. Garg *et al.* mention that what is considered relevant tag to a given picture (and just the picture without any context or background information) can vary from user to user due to difference in perception [60]. We agree that validation by few users is not perfect but we also believe that accuracy results based on manual validation can still provide useful insight into the quality of tag recommendation techniques. Hence, user validation can aid evaluation of automatic tag recommendation systems.

Figure 8.3: Snapshot of photos associated with tweets to illustrate semantic enhancement of images using tweet content

In this technique, we ask annotators to check if tags recommended by our system *semantically enhance* the information in the associated multimedia object. A tag semantically enhances the associated media object if it provides more information than currently present tags, title and description in the media object. Figure 8.3 shows some examples describing positive and negative cases of semantic enhancement.

Evaluation Metrics

In order to formalize the evaluation metrics – Overlap Count, Average Overlap Percentage and Relevance Score we first define the following.

$$
\begin{aligned}
HT_i &= \text{Set of Hashtags in Tweet}_i \\
T_i &= \text{Set of media tags corresponding to Tweet}_i \\
TT_i &= \text{Set of tokens in Tweet}_i \\
N &= \text{Total number of Tweets}
\end{aligned}
$$

We now define evaluation metrics for each of the two validation techniques viz. automated and manual.

Automated

We use two metrics to evaluate our tag recommendation system – Overlap Count (OC) and Average Overlap Percentage (AOP). We define them as follows:

Overlap Count (OC) Overlap Count measures the number of overlaps of tags recommended by our system with the tags present in the associated media object.

$$\text{OC} = \Sigma_{i=1}^{N} |HT_i \cap T_i|$$

Table 8.5: Experimental Dataset (X= { Y: YouTube, D : Dailymotion, F : Flickr, P: Photobucket, S : SoundCloud} , DX_i denotes number of tweets containing URLs of X to answer RQ_i, $\forall i = \{1, 2\}$)

		Validation Technique	
Service Media	**No. of tweets**	**Automated**	**Manual**
YouTube	DY1	656	14
	DY2	656 + 1566 = 2222	14 + 44 = 58
Dailymotion	DD1	65	5
	DD2	65 + 196 = 261	5 + 24 = 29
Flickr	DF1	613	160
	DF2	613 + 1470 = 2083	160 + 590 = 750
Photobucket	DP1	24	92
	DP2	24 + 569 = 593	92 + 802 = 894
SoundCloud	DS1	189	118
	DS2	189 + 2944 = 3133	118 + 2545 = 2663

Average Overlap Percentage (AOP) In order to calculate Average Overlap Percentage, we first calculate OC of each tweet per number of tokens in that tweet (TT_i). We then take percentage average across all tweets.

$$\text{AOP} = \left(\Sigma_{i=1}^{N} \left(\frac{|HT_i \cap T_i|}{|TT_i|} \right) / N \right) * 100$$

Manual

In the manual validation technique, we use Relevance Score (RS) to evaluate our tag recommendation system. We define them as follows:

Relevance Score (RS) Relevance Score measures the average number of relevant tags by our tag recommendation system with respect to the associated multimedia object.

$$R_i = \begin{cases} 1 & \text{if content of Tweet}_i \text{ is marked} \\ & \text{relevant by annotators} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{RS} = \frac{\Sigma_{i=1}^{N} R_i}{N}$$

8.5 Results

In order to address the research questions given in Section 8.2 with appropriate validation techniques (automated and manual), we divide our dataset into four buckets for each media service. Table 8.5 describes the number of tweets present in each bucket for each media service and corresponding validation technique. We use the following four cases to detail the same:

* **Case I : RQ1-Automated** In this case, we consider those tweets which contain hashtags and the corresponding media also contains tags. In Table 8.5, {DY1, DD1, DF1, DP1, DS1}–Automated cells represent the number of tweets corresponding to this case. Example I of Table 8.1 illustrates this case. We perform *automated* validation as discussed in Section 8.4. We report our results on the *overlap count* and *average overlap score* metrics for this case.

* **Case II : RQ2-Automated** In this case, we consider those tweets which may or may not contain hashtags but the corresponding media contains tags. In Table 8.5, {DY2, DD2, DF2, DP2, DS2}–Automated cells represent the number of tweets corresponding to this case. Example II of Table 8.1 illustrates this case. We perform *automated* validation as discussed in Section 8.4. We report our results on the *overlap count* and *average overlap score* metrics for this case.

* **Case III: RQ1-Manual** In this case, we consider those tweets which contain hashtags but the corresponding media doesn't contain tags. In Table 8.5, {DY1, DD1, DF1, DP1, DS1}–Manual cells represent the number of tweets corresponding to this case. Example III of Table 8.1 illustrates this case. We perform *manual* validation as discussed in Section 8.4. We report our results on the *relevance score* metric for this case.

* **Case IV: RQ2-Manual** In this case, we consider those tweets which may or may not contain contain hashtags and the corresponding media doesn't contain tags. In Table 8.5, {DY2, DD2, DF2, DP2, DS2}–Manual cells represent the number of tweets corresponding to this case. Example IV of Table 8.1 illustrates this case. We perform *manual* validation as discussed in Section 8.4. We report our results on the *relevance score* metric for this case.

Table 8.6 shows results on evaluation metrics (OC, AOP) for the automated validation technique and Table 8.7 shows results on evaluation metric (RS) for the manual validation technique.

The Overlap Count and Average Overlap Scores in Table 8.6 suggest that there's a high amount of overlap between tweet content (with or without hashtags) and the associated media tags. The Relevance Scores in Table 8.7 demonstrate that tweet content (with or without hashtags) can be used to semantically annotate and enrich associated multimedia objects.

8.6 Discussion

Our solution employs a Wisdom of Crowd mashup framework for social tag recommendation. Our solution is language independent and hence, can be used to recommend multi-lingual tags on

Table 8.6: Evaluation Metrics for Automated Validation Technique (OC : Overlap Count, AOP : Average Overlap Percentage)

		Evaluation Metric	
Service Media		**Overlap Count (OC) / Number of Tweets**	**Average Overlap Percentage (AOP)**
YouTube	RQ1	133/656	1.92
	RQ2	1883/2222	5.17
Dailymotion	RQ1	12/65	1.88
	RQ2	360/261	17.71
Flickr	RQ1	420/613	7.78
	RQ2	1580/2083	5.58
Photobucket	RQ1	17/24	9.45
	RQ2	530/593	17.14
SoundCloud	RQ1	18/189	0.89
	RQ2	4508/3133	15.41

Table 8.7: Evaluation Metrics for Manual Validation Technique

	Relevance Score				
	YouTube	**Dailymotion**	**Flickr**	**Photobucket**	**SoundCloud**
RQ1	85.71%	40 %	79 %	40.91 %	57.84 %
RQ2	84.48%	29.62%	53.86%	37.02 %	62.59 %

social media services. Our tag recommendation system could be used to enhance Information Retrieval systems like news recommendation, online shopping and personalization of online content. However, a more sophisticated algorithm to selectively choose content from tweets as tags could be used to improve the quality of recommended tags. Also, the coverage our system would be high for multimedia objects only on external social media services which are frequently shared on Twitter. Hence, our technique is a complementary approach to the existing work on automatic tag recommendation systems.

8.7 Conclusion

We present an approach to automatically recommend tags on social media services. The proposed approach employs a *crowdsourcing* framework by mining Twitter to recommend tags on image, video and audio sharing services like Flickr, Photobucket, Dailymotion, YouTube and SoundCloud. We use a hybrid technique to perform validation and propose appropriate evaluation metrics corresponding to the techniques. The results show that content of tweets can be used to recommend tags on various social media services.

Chapter 9

Summary

In this chapter, we conclude by providing a brief summary of the contributions in context with the thesis objectives. In this thesis, we have three objectives with respect to content quality on Web 2.0 services – (1) Low Quality Content, (2) Content Quality Systems and (3) Information Retrieval. We summarize the specific contributions across each of these aspects.

9.1 Low Quality Content

We look at low quality questions on Stackoverflow - a popular programming based CQA site.

1. We look at *closed* questions on Stackoverflow which are by definition deemed unfit for its Q&A format and hence, low quality. We analyze these questions and try to understand various aspects like community participation and temporal patterns. We also build a machine learning based system to detect a *closed* question at question creation time based on profile, stylistic, community process and content based features.

2. We also look at *deleted* questions on Stackoverflow which are removed from the site for being very poor in quality. We analyze such questions for aspects like user deletion patterns and discover a pyramidal structure of question quality. We also build a machine learning based system to detect a *deleted* question at creation time based on variety of features.

We address the problem of low quality content on collaborative crowdsourced question answering websites and try to build systems to detect such content to help the users and gatekeepers of the community. In both studies, we make data publicly available for research purposes.

9.2 Content Quality Systems

In both the previous studies, we make the two critical observations – (1) content quality has different levels of "poorness" factors and (2) content quality is directly linked to characteristics involving

daily work flow. In this part, we look to improve the daily workflow of the user and hence, enhance user experience which in turn helps the user to contribute better quality content. Therefore, We build two systems – CQA recommendation enginge and a Web reference management browser plugin – to help improve quality on Issue Tracking Systems used during the software maintenance process - Issue Trackers.

1. We perform a survey of software maintenance professionals from two large and popular open source projects – Google Chromium and Android. We motivate the need of CQA usage by looking at discussions in Issue Tracking Systems. We build a recommendation system to suggest relevant CQA questions in an Issue Tracker System thus, reducing the context switch for developers during maintenance.

2. We conduct a survey to understand web browsing patterns of software developers during the maintenance process. We find that bug fixers frequently use the Internet to search for references and use them during daily tasks. We analyze such references to gain insights. In addition, we build a web browser plugin – *Samekana* – to help software maintenance professionals manage references in an Issue Tracking System. The browser plugin is publicly available to download and use.

Overall, we propose some solutions to enhance content quality on Issue Tracking Systems – an artifact which is regularly used during the software maintenance process. We plan to make our data publicly available for research purposes.

9.3 Information Retrieval Enhancement

In this part, we look at the IR aspect of content quality. Specifically, we look at two aspects to enhance IR - (1) Tag recommendation to enhance metadata of content and (2) Utilization of quality content to discover social communities.

1. We recognize that tags are an important source of information to enable Information Retrieval on social media. We propose an alternative recommendation approach to suggest tags on social media. Specifically, we look into the use of Twitter to suggest tags for linked photos, images etc. on external social media sites.

2. We utilize interactions made by users on a social network to discover topic-specific communities. We look at a memory mindful solution that does not require the entire social network graph to be loaded into memory. We test our approach on Twitter and show that interactions are useful quality enablers to discover implicit topic-based communities.

In this contribution, we investigate novel approaches to enhance quality on social media to aid Information Retrieval tasks. We look into two quality indiciators – *tags* and *social interactions* – and show the usefulness of our approach.

Bibliography

[1] Android issue tracker. https://code.google.com/p/android/issues/list.

[2] Chromium issue tracker. https://code.google.com/p/chromium/issues/list.

[3] Facebook. https://www.facebook.com.

[4] Flickr. https://www.flickr.com.

[5] Foursquare. https://foursquare.com.

[6] Privileges - create tags. http://stackoverflow.com/privileges/create-tags.

[7] Quora. http://quora.com.

[8] Stack exchange data explorer. https://data.stackexchange.com.

[9] Stack overflow. http://stackoverflow.com.

[10] Twitter. https://twitter.com.

[11] Why and how are some questions deleted? http://stackoverflow.com/help/deleted-questions.

[12] Why are some questions closed, and what does ”closed” mean? http://stackoverflow.com/help/closed-questions.

[13] Yahoo! answers. https://answers.yahoo.com.

[14] Youtube. https://www.youtube.com.

[15] How does deleting work? what can cause a post to be deleted, and what does that actually mean? what are the criteria for deletion? http://meta.stackoverflow.com/q/5221/214223, September 2008.

[16] To outdo google, naver taps into korea's collective wisdom to outdo google, naver taps into korea's collective wisdom. http://www.nytimes.com/2007/07/04/technology/04iht-naver.2.6485099.html?pagewanted=all&_r=0, September 2008.

[17] What are "community wiki" posts? http://meta.stackoverflow.com/questions/11740/what-are-community-wiki-posts, September 2008.

[18] What is a "locked" post? http://meta.stackoverflow.com/questions/22228/what-is-a-locked-post, September 2008.

[19] The great question deletion audit of 2010. http://meta.stackoverflow.com/questions/51097/the-great-question-deletion-audit-of-2010, May 2010.

[20] What is a "protected" question? http://meta.stackoverflow.com/questions/52764/what-is-a-protected-question/, June 2010.

[21] Who are the diamond moderators, and what is their role? http://meta.stackoverflow.com/a/75192/214223, January 2011.

[22] Stack exchange data dump. http://www.clearbits.net/torrents/2076-aug-2012, August 2012.

[23] List of stack exchange moderators by sites. http://stackexchange.com/about/moderators?by=sites, June 2013.

[24] Twitter ipo filing. http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htm, October 2013.

[25] What is a day in life of a stackoverflow moderator? http://meta.stackoverflow.com/a/166630/214223, February 2013.

[26] Anupama Aggarwal, Ashwin Rajadesingan, and Ponnurangam Kumaraguru. Phishari: Automatic realtime phishing detection on twitter. In *eCrime Researchers Summit (eCrime), 2012*, pages 1–12. IEEE, 2012.

[27] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the international conference on Web search and web data mining*, pages 183–194. ACM, 2008.

[28] Miltiadis Allamanis and Charles Sutton. Why, when, and what: analyzing stack overflow questions by topic, type, and code. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 53–56. IEEE Press, 2013.

[29] Reid Andersen and Kevin J. Lang. Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 223–232, New York, NY, USA, 2006. ACM.

[30] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 850–858. ACM, 2012.

[31] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 850–858, New York, NY, USA, 2012. ACM.

[32] Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. Steering user behavior with badges. 2013.

[33] Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K Roy, and Kevin A Schneider. Answering questions about unanswered questions of stack overflow. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 97–100. IEEE Press, 2013.

[34] Jeff Atwood. Stack overflow creative commons data dump. `http://blog.stackoverflow.com/2009/06/stack-overflow-creative-commons-data-dump/`, June 2009.

[35] Alberto Bacchelli, Luca Ponzanelli, and Michele Lanza. Harnessing stack overflow for the ide. In *3rd International Workshop on Recommendation Systems for Software Engineering*, RSSE '12, 2012.

[36] Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 44–54, New York, NY, USA, 2006. ACM.

[37] Ricardo A Baeza-Yates and Luz Rello. How bad do you spell?: The lexical quality of social media. In *The Future of the Social Web*, 2011.

[38] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, pages 1–36, 2012.

[39] Grigory Begelman, Philipp Keller, and Frank Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, 2006.

[40] Dane Bertram, Amy Voida, Saul Greenberg, and Robert Walker. Communication, collaboration, and bugs: the social nature of issue tracking in small, collocated teams. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW '10, pages 291–300, New York, NY, USA, 2010. ACM.

[41] Mohan John Blooma, Dion Hoe-Lian Goh, and Alton Yeow-Kuan Chua. Predictors of high-quality answers. *Online Information Review*, 36(3):383–400, 2012.

[42] Stephen P Borgatti and Martin G Everett. Models of core/periphery structures. *Social Networks*, 21(4):375 – 395, 2000.

[43] Amiangshu Bosu, Christopher S Corley, Dustin Heaton, Debarshi Chatterji, Jeffrey C Carver, and Nicholas A Kraft. Building reputation in stackoverflow: an empirical investigation. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 89–92. IEEE Press, 2013.

[44] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. Example-centric programming: integrating web search into the development environment. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 513–522, New York, NY, USA, 2010. ACM.

[45] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1589–1598, New York, NY, USA, 2009. ACM.

[46] Grégoire Burel, Yulan He, and Harith Alani. Automatic identification of best answers in online enquiry communities. In *The Semantic Web: Research and Applications*, pages 514–529. Springer, 2012.

[47] Yuanzhe Cai and Sharma Chakravarthy. Predicting answer quality in q/a social networks: Using temporal features.

[48] Joshua Charles Campbell, Chenlei Zhang, Zhen Xu, Abram Hindle, and James Miller. Deficient documentation detection: a methodology to locate deficient project documentation using topic analysis. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 57–60. IEEE Press, 2013.

[49] Sidharth Chhabra, Anupama Aggarwal, Fabricio Benevenuto, and Ponnurangam Kumaraguru. Phi. sh$ ocial: the phishing landscape through short urls. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, pages 92–101. ACM, 2011.

[50] Aaron Clauset. Finding local community structure in networks. *Phys. Rev. E*, 72:026132, Aug 2005.

[51] Joel Cordeiro, Bruno Antunes, and Paulo Gomes. Context-based recommendation to support problem solving in software development. In *3rd International Workshop on Recommendation Systems for Software Engineering*, RSSE '12, 2012.

[52] Denzil Correa, Sangeeta Lal, Apoorv Saini, and Ashish Sureka. Samekana: A browser extension for including relevant web links in issue tracking system discussion forum.

[53] Denzil Correa and Ashish Sureka. Fit or unfit: analysis and prediction of 'closed questions' on stack overflow. In *Proceedings of the first ACM conference on Online social networks*, COSN '13, pages 201–212, New York, NY, USA, 2013. ACM.

[54] Denzil Correa and Ashish Sureka. Integrating issue tracking systems with community-based question and answering websites. In *Proceedings of the 22nd Australasian Software Engineering Conference*, ASWEC '13. IEEE Press, 2013.

[55] Bogdan Dit and Andrian Marcus. Improving the readability of defect reports. In *Proceedings of the 2008 international workshop on Recommendation systems for software engineering*, RSSE '08, pages 47–49, New York, NY, USA, 2008. ACM.

[56] Pnina Fichman. A comparative assessment of answer quality on four question answering sites. *Journal of Information Science*, 37(5):476–486, 2011.

[57] Adam Fourney and Meredith Ringel Morris. Enhancing technical q&a forums with citehistory. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, ICWSM '13, 2013.

[58] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[59] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[60] Nikhil Garg and Ingmar Weber. Personalized, interactive tag recommendation for flickr. In *ACM conference on Recommender systems*, RecSys '08, pages 67–74, New York, NY, USA, 2008. ACM.

[61] R Stuart Geiger, Aaron Halfaker, Maryana Pinchuk, and Steven Walling. Defense mechanism or socialization tactic? improving. 2012.

[62] Alexandru Lucian Ginsca and Adrian Popescu. User profiling for answer quality assessment in q&a communities. In *Proceedings of the 2103 workshop on Data-driven user behavioral modelling and mining from social media*, pages 25–28. ACM, 2013.

[63] Max Goldman and Robert C. Miller. Codetrail: Connecting source code and web resources. *Visual Languages and Human-Centric Computing, IEEE Symposium on*, 0:65–72, 2008.

[64] Carlos Gómez, Brendan Cleary, and Leif Singer. A study of innovation diffusion through link sharing on stack overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 81–84, Piscataway, NJ, USA, 2013. IEEE Press.

[65] Carlos Gómez, Brendan Cleary, and Leif Singer. A study of innovation diffusion through link sharing on stack overflow. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 81–84. IEEE Press, 2013.

[66] B Goncalves, N Perra, and A Vespignani. Modeling users' activity on twitter networks: validation of dunbar's number. *PloS one*, 6(8), 2011.

[67] Swapna Gottipati, David Lo, and Jing Jiang. Finding relevant answers in software forums. *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, 0:323–332, 2011.

[68] F Maxwell Harper, Daphne Raban, Sheizaf Rafaeli, and Joseph A Konstan. Predictors of answer quality in online q&a sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 865–874. ACM, 2008.

[69] Björn Hartmann, Mark Dhillon, and Matthew K. Chan. Hypersource: bridging the gap between source and code-related web sites. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2207–2210, New York, NY, USA, 2011. ACM.

[70] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[71] Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. Fighting spam on social web sites: A survey of approaches and future challenges. *Internet Computing, IEEE*, 11(6):36–45, 2007.

[72] S. Hill, F. Provost, and C. Volinsky. Network-Based Marketing: Identifying Likely Adopters via Consumer Networks. *ArXiv Mathematics e-prints*, June 2006.

[73] Haifeng Hu, Bingquan Liu, Baoxun Wang, Ming Liu, and Xiaolong Wang. Multimodal dbn for predicting high-quality answers in cqa portals.

[74] B. A. Huberman, D. M. Romero, and F. Wu. Social networks that matter: Twitter under the microscope. *ArXiv e-prints*, December 2008.

[75] Joel Spolsky Jeff Atwood. Stack exchange platform. `http://stackexchange.com`, September 2009.

[76] Jiwoon Jeon, W. Bruce Croft, Joon Ho Lee, and Soyeon Park. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 228–235, New York, NY, USA, 2006. ACM.

[77] Sascha Just, Rahul Premraj, and Thomas Zimmermann. Towards the next generation of bug tracking systems. *Visual Languages and Human-Centric Computing, IEEE Symposium on*, 0:82–85, 2008.

[78] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *ECML/PKDD 2008 Discovery Challenge*, 2008.

[79] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 137–146, New York, NY, USA, 2003. ACM.

[80] Lyndon S. Kennedy, Shih-Fu Chang, and Igor V. Kozintsev. To search or to label?: predicting the performance of search-based automatic image classifiers. In *MIR*, MIR '06, pages 249–258, New York, NY, USA, 2006. ACM.

[81] Mohit Kewalramani. Community Detection in Twitter. Master's thesis, May 2011.

[82] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. *Computer Networks*, 31:1481 – 1493, 1999.

[83] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 591–600, New York, NY, USA, 2010. ACM.

[84] Kyle Lacy. *Twitter Marketing for Dummies*, volume 2nd. 2009.

[85] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers: social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442. ACM, 2010.

[86] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 695–704, New York, NY, USA, 2008. ACM.

[87] Baichuan Li, Tan Jin, Michael R. Lyu, Irwin King, and Barley Mak. Analyzing and predicting question quality in community question answering services. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 775–782, New York, NY, USA, 2012. ACM.

[88] Mario Linares-Vásquez, Bogdan Dit, and Denys Poshyvanyk. An exploratory analysis of mobile development issues using stack overflow. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 93–96. IEEE Press, 2013.

[89] Dong Liu, Xian-Sheng Hua, Linjun Yang, Meng Wang, and Hong-Jiang Zhang. Tag ranking. In *WWW*, WWW '09, pages 351–360, New York, NY, USA, 2009. ACM.

[90] Rafael Lotufo, Zeeshan Malik, and Krzysztof Czarnecki.

[91] Rafael Lotufo, Zeeshan Malik, and Krzysztof Czarnecki. Modelling the 'hurried' bug report reading process to summarize bug reports. In *ICSM*, pages 430–439. IEEE Computer Society, 2012.

[92] Rafael Lotufo, Leonardo Passos, and Krzysztof Czarnecki. Towards improving bug tracking systems with game mechanisms. In *9th Working Conference on Mining Software Repositories (MSR'12)*, Zurich, Switzerland, 06/2012 2012. IEEE (also published as GSDLAB–TR 2011–09–29), IEEE (also published as GSDLAB–TR 2011–09–29).

[93] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2857–2866. ACM, 2011.

[94] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest q&a site in the west. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2857–2866, New York, NY, USA, 2011. ACM.

[95] E. Moxley, T. Mei, and B. S. Manjunath. Video annotation through search and graph reinforcement mining. *IEEE Transactions on Multimedia*, 12(3):184–193, Apr 2010.

[96] Seyed Mehdi Nasehi, Jonathan Sillito, Frank Maurer, and Chris Burns. What makes a good code example?: A study of programming q&a in stackoverflow. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 25–34. IEEE, 2012.

[97] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, Jun 2004.

[98] M. E. J. Newman. From the Cover: Modularity and community structure in networks. *Proceedings of the National Academy of Science*, 103:8577–8582, June 2006.

[99] Dor Nir, Shmuel Tyszberowicz, and Amiram Yehudai. Locating regression bugs. In *Hardware and Software: Verification and Testing*, pages 218–234. Springer, 2008.

[100] Simon Overell, Börkur Sigurbjörnsson, and Roelof van Zwol. Classifying tags using open content resources. In *WSDM*, WSDM '09, pages 64–73, New York, NY, USA, 2009. ACM.

[101] Aditya Pal, F Maxwell Harper, and Joseph A Konstan. Exploring question selection bias to identify experts and potential experts in community question answering. *ACM Transactions on Information Systems (TOIS)*, 30(2):10, 2012.

[102] S. Papadopoulos, A. Skusa, A. Vakali, Y. Kompatsiaris, and N. Wagner. Bridge Bounding: A Local Approach for Efficient Community Discovery in Complex Networks. *ArXiv e-prints*, February 2009.

[103] Chris Parnin, Christoph Treude, Lars Grammel, and Margaret-Anne Storey. Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow. *Georgia Institute of Technology, Tech. Rep*.

[104] Nishith Pathak, Colin Delong, Arindam Banerjee, and Kendrick Erickson. Social Topic Models for Community Extraction. In *The 2nd SNA-KDD Workshop '08 (SNA-KDD'08)*, August 2008.

[105] Luca Ponzanelli, Alberto Bacchelli, and Michele Lanza. Seahawk: stack overflow in the ide. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 1295–1298. IEEE Press, 2013.

[106] Sarah Rastkar, Gail C. Murphy, and Gabriel Murray. Summarizing software artifacts: a case study of bug reports. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 505–514, New York, NY, USA, 2010. ACM.

[107] R.I.M. and Dunbar. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 22(6):469 – 493, 1992.

[108] Mrinmaya Sachan, Danish Contractor, Tanveer Faruquie, and Venkata Subramaniam. Probabilistic model for discovering topic based communities in social networks. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 2349–2352, New York, NY, USA, 2011. ACM.

[109] Mrinmaya Sachan, Danish Contractor, Tanveer A. Faruquie, and L. Venkata Subramaniam. Using content and interactions for discovering communities in social networks. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 331–340, New York, NY, USA, 2012. ACM.

[110] Tetsuya Sakai, Daisuke Ishikawa, Noriko Kando, Yohei Seki, Kazuko Kuriyama, and Chin-Yew Lin. Using graded-relevance metrics for evaluating community qa answer selection. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 187–196. ACM, 2011.

[111] Santo and Fortunato. Community detection in graphs. *Physics Reports*, 486:75 – 174, 2010.

[112] Nicholas Sawadsky and Gail C. Murphy. Fishtail: from task context to source code examples. In *Proceedings of the 1st Workshop on Developing Tools as Plug-ins*, TOPI '11, pages 48–51, New York, NY, USA, 2011. ACM.

[113] Chirag Shah and Jefferey Pomerantz. Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 411–418. ACM, 2010.

[114] Börkur Sigurbjörnsson and Roelof van Zwol. Flickr tag recommendation based on collective knowledge. In *WWW*, pages 327–336, New York, NY, USA, 2008. ACM.

[115] Yang Song, Ziming Zhuang, Huajing Li, Qiankun Zhao, Jia Li, Wang-Chien Lee, and C. Lee Giles. Real-time automatic tag recommendation. In *SIGIR*, SIGIR '08, pages 515–522, New York, NY, USA, 2008. ACM.

[116] Sanjay C. Sood, Sara H. Owsley, Kristian J. Hammond, and Larry Birnbaum. Tagassist: Automatic tag suggestion for blog posts. 2007.

[117] Avaré Stewart, Ernesto Diaz-Aviles, Wolfgang Nejdl, Leandro Balby Marinho, Alexandros Nanopoulos, and Schmidt-Thie Lars. Cross-tagging for personalized open social networking. In *ACM conference on Hypertext and hypermedia*, HT '09, pages 271–278, New York, NY, USA, 2009. ACM.

[118] Ashish Sureka, Ponnurangam Kumaraguru, Atul Goyal, and Sidharth Chhabra. Mining youtube to discover extremist videos, users and hidden communities. In *Information Retrieval Technology*, volume 6458 of *Lecture Notes in Computer Science*, pages 13–24. Springer Berlin / Heidelberg, 2010.

[119] Lei Tang and Huan Liu. Graph mining applications to social network analysis. In Charu C. Aggarwal, Haixun Wang, and Ahmed K. Elmagarmid, editors, *Managing and Mining Graph Data*, volume 40 of *The Kluwer International Series on Advances in Database Systems*, pages 487–513. Springer US, 2010.

[120] Lei Tang, Huan Liu, Jianping Zhang, and Zohreh Nazeri. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 677–685, New York, NY, USA, 2008. ACM.

[121] Lei Tang, Xufei Wang, and Huan Liu. Uncovering groups via heterogeneous interaction analysis. In *ICDM '09: Proceedings of IEEE International Conference on Data Mining*, pages 503–512, 2009.

[122] Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 717–726, New York, NY, USA, 2007. ACM.

[123] Yla R Tausczik and James W Pennebaker. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24–54, 2010.

[124] Jaime Teevan, Daniel Ramage, and Merredith Ringel Morris. # twittersearch: a comparison of microblog search and web search. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 35–44. ACM, 2011.

[125] Hapnes Toba, Zhao-Yan Ming, Mirna Adriani, and Tat-Seng Chua. Discovering high quality answers in community question answering archives using a hierarchy of classifiers. *Information Sciences*, 261:101–115, 2014.

[126] G. Toderici, H. Aradhye, M. Pasca, L. Sbaiz, and J. Yagnik. Finding Meaning on YouTube: Tag recommendation and Category Discovery. In *CVPR'10*, 2010.

[127] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (nier track). In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 804–807, New York, NY, USA, 2011. ACM.

[128] G. Tscherteu and C. Langreiter. Community core detection in twitter - a "bottom up" heuristic.

[129] Hans Van Vliet, Hans Van Vliet, and JC Van Vliet. *Software engineering: principles and practice*, volume 3. Wiley, 1993.

[130] Ken Wakita and Toshiyuki Tsurumi. Finding community structure in mega-scale social networks: [extended abstract]. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1275–1276, New York, NY, USA, 2007. ACM.

[131] Wei Wang and Michael W Godfrey. Detecting api usage obstacles: a study of ios and android developer questions. In *Proceedings of the Tenth International Workshop on Mining Software Repositories*, pages 61–64. IEEE Press, 2013.

[132] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. A bayesian approach toward finding communities and their evolutions in dynamic social networks. In *SDM'09*, pages 990–1001, 2009.

[133] Tianbao Yang, Rong Jin, Yun Chi, and Shenghuo Zhu. Combining link and content for community detection: a discriminative approach. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 927–936, New York, NY, USA, 2009. ACM.

[134] Yuan Yao, Hanghang Tong, Tao Xie, Leman Akoglu, Feng Xu, and Jian Lu. Want a good answer? ask a good question first! *arXiv preprint arXiv:1311.6876*, 2013.

[135] Alexey Zagalsky, Ohad Barzilay, and Amiram Yehudai. Example overflow: Using social media for code recommendation. In *3rd International Workshop on Recommendation Systems for Software Engineering*, RSSE '12, 2012.

[136] Wan-Lei Zhao, Xiao Wu, and Chong-Wah Ngo. On the annotation of web videos by efficient near-duplicate search. *IEEE Transactions on Multimedia*, 12(5):448–461, 2010.

[137] Guangyou Zhou, Kang Liu, and Jun Zhao. Joint relevance and answer quality learning for question routing in community qa. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1492–1496. ACM, 2012.

[138] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost. *Ann Arbor*, 1001(48109):1612, 2006.

[139] Jinfeng Zhuang and Steven C.H. Hoi. A two-view learning approach for image tag ranking. In *WSDM*, WSDM '11, pages 625–634, New York, NY, USA, 2011. ACM.

[140] Thomas Zimmermann, Rahul Premraj, Jonathan Sillito, and Silvia Breu. Improving bug tracking systems. In *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Companion Volume*, pages 247–250. IEEE, 2009.