Expert identification in the ggplot2 community

Demmers, J.

Award date:
2015

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Software Engineering and Technology Research Group

Expert identification in the ggplot2 community

Master Thesis

Jos Demmers

Supervisor:
dr. A. Serebrenik
Committee:
dr. R. Kuiper
dr. M. Pechenizkiy

final version

Eindhoven - December 14, 2015

Abstract

This research project will analyse Open Source and Questions and Answers (Q&A) communities to detect experts. This project aims at assisting Q&A communities in finding experts for unanswered questions. Furthermore this research project will help Open Source projects on Github by introducing a way to find experts that can help with unsolved issues and discussions. Identified experts can either give extra feedback on issues, help with minimal reproducible examples or solve it themselves, depending on their knowledge. By using data mining techniques information such as user names, email addresses, and user contributions will be extracted from GitHub, Stack Overflow, and Google Groups. By using the user contribution data, a general and focussed level of expertise is measured for each user. In this project we will consider ggplot2 as a case study, ggplot2 is a plotting system for R for creating statistical graphics.

Preface

This master thesis is the result of my graduation project which completes my Computer Science and Engineering study at Eindhoven University of Technology. The project was performed internally at the Software Engineering and Technology group of the Mathematics and Computer Science department of Eindhoven University of Technology. First of all I would like to thank my supervisor, Alexander, for the weekly meetings and great feedback during my thesis. I would also like to thank the committee members for their interest in my thesis. Finally my thanks to the ggplot2 community at Google Groups for their interest and participation in the surveys.

Contents

List of Figures

List of Tables

List of Code

Chapter 1

Introduction

Nowadays there are many Q&A communities, some of them accepting a wide variety of topics while other communities only focus on a specific topic or product. Communities like 'Quora'[1], 'Answers'[2], 'Yahoo! Answers'[3], and 'Google Groups'[4] cover a broad range of topics. The topic-specific communities can vary greatly, e.g. 'Grammarly'[5], Q&A site on English grammar and usage, 'TurboTax Live Community'[6], Q&A site for questions related to tax and TurboTax products. Furthermore the Stack Exchange[7] network contains more than 150 Q&A communities on a variety of topics. Some of them focus on technology, others on culture, recreation, life, arts, science, business, and many more. Regarding technology, programming to be more specific, the Stack Overflow[8] community is very popular, assuming popularity is indicated by the amount questions, answers, and users. Since the launch in 2008 the community has grown significantly and currently has a user-base of more than 4.6 million users. Back in 2010 Mamykina et al.[22] reported a user-base of approximately 0.3 million, that is an increase of more than 1500% over the past 5 years. At that time the median answer time of questions was only 11 minutes. As of August 2015 the number of programming related questions has passed the 10 million mark[9], and the total number of given solutions is more than 16,5 million. Furthermore the developers active in the community currently post an answer every 12 seconds. Another site worth mentioning is Google Groups, which we also consider a Q&A community. It offers a discussion framework and everyone is free to create their own discussion group. Google Groups works by using mailing lists although communities are also accessible through a web interface. Mailing lists have had multiple purposes, e.g. patch distributions, or newsletters, however on Google Groups a mailing list is used for Q&A and discussion purposes.

Besides programming related Q&A communities there are also communities for code hosting, e.g GitHub[10], GitLab[11], and Sourceforge[12]. Both type of communities are related to each other as users from Q&A communities put their expertise into practice in the code hosting communities and users encountering issues in various projects will look for solutions in the Q&A communities. Advantages of code hosting communities are that they make collaboration easier, provide an easy to access backup to projects and have interesting offers to attract new users[13]. A popular example is GitHub which has been used in multiple studies [12][23][37][38]. GitHub is social like Stack

[1] https://www.quora.com
[2] http://www.answers.com
[3] https://answers.yahoo.com
[4] https://groups.google.com
[5] http://www.grammarly.com/answers
[6] https://ttlc.intuit.com
[7] http://stackexchange.com/sites
[8] http://stackoverflow.com
[9] http://stackoverflow.com/10m
[10] https://github.com
[11] https://gitlab.com/explore
[12] http://sourceforge.net
[13] https://github.com/blog/1900-the-best-developer-tools-now-free-for-students

Overflow, has a user-base of more than 10 million, and more than 26 million repositories. Social aspects of GitHub are that each user has a profile, users or specific projects can be followed to keep up-to-date, and users can collaborate on projects with each other and discuss project issues that arise during the development.

There are a wide variety of users in online communities like Stack Overflow, Google Groups, and GitHub, we are interested in the experts. Expert users are of importance because they play an important role in online communities, they create high quality answers which in return may attract new users. Identifying experts is also necessary to suggest experts for unsolved questions and issues. Besides this we are interested in identifying experts because that information is helpful for others. For example for bug triaging it could assist in assigning developers to bug reports by selecting developers that have the required expertise. In addition members of Q&A communities could use this information when they have a very specific question to ask the right expert. An interesting aspect for recruiters is the possibility to create expertise profiles of users. Previous studies indicate that a relatively small number of users, the experts, are responsible for the majority of answers and commits [15][26][27]. Often it is however rather difficult to find such experts and to say what their topic(s) of expertise are. The small group of experts are difficult to distinguish in communities varying from thousands to millions of users. Stack Overflow does have some indicators to distinguish experts with their reputation and tagging system although the tags can be too general when you are looking for an expert on a specific topic. Google Groups on the other hand does not have any indicators at all. To be able to suggest experts for unsolved questions and issues each user needs to be thoroughly analysed to find their area of expertise. Previous studies have used a number of different techniques to identify experts, e.g., analysing user behaviour [26][28], term/word frequencies [38], source code contribution [35][29]. However, using a single source of information and focussing on one technique has its limitations, as indicated by Littlepage et al. [19] who addresses the limitations of term/word frequencies. Furthermore the advantage of using multiple sources is a broader view on the expertise of users.

1.1 Project Goal

This thesis combines a number of existing techniques on expertise finding, we can group them in two different categories. First of all the general methods based on activity, i.e. number of questions and answers. Secondly the more detailed methods that analyse users' contributions to be able to find their topics of expertise using information retrieval techniques. By combining both methods we believe that it will result in a good indication of a user's expertise. To validate our findings we will use both automated and manual tests. For the manual tests surveys are created that are filled in by community members and are used to verify the expertise rankings and the topics of expertise for users.

The expertise ranking algorithms will be implemented in a case study, for which we use ggplot2. Ggplot2 is a project on GitHub and is a plotting system for R for creating statistical graphics. It takes care of many steps and difficulties that you encounter with plotting, e.g. drawing legends, statistical transformations of data, and scaling. Ggplot2 is chosen for this case study because of its active development, size and community. In addition we are familiar with ggplot2 as we have used the R package multiple times in varies projects. The project is active on GitHub since November 2007, has around 60 contributors and a total of 3100 commits. There are also active discussions on Stack Overflow and Google Groups, almost 5000 questions have been asked on Google Groups and on Stack Overflow more than 6000. Combined those communities have more than enough data to identify and find experts but not too much to lose track of it.

The novelty of this approach is that most current work on expert finding focusses on a single source of data, i.e. a single project or community. The expertise data is then used within that same community to recommend experts for specific problems. However, users are often active in multiple communities and expertise data gathered in one community could be useful in other communities as well. Two recent works that use expertise data across communities as well are those of Venkataramani et al. [38] and Badashian et al. [6]. Venkataramani et al. uses GitHub

data to recommend experts for unanswered questions on Stack Overflow and Badashian et al. uses Stack Overflow to recommend experts for GitHub issues. In contrast to those works we do not use a single source to find experts but combine the data of Stack Overflow and Google Groups. Furthermore the use of expertise information is not limited to those communities but can be used for GitHub as well. Both works are discussed in more detail in Related Work, Chapter 5.

For this case study multiple expertise ranking algorithms will be implemented and compared varying from basic to more advanced. The results will be used to test what method is most suitable to identify experts in our use case. The identifying of experts will be implemented using an iterative approach, starting with a general approach that focusses on users activity, i.e. amount of questions and answers, but does not take into account the ggplot2 features a user is familiar with, we refer to this approach as general expertise. The next iteration applies textual analysis on user's posts in the Stack Overflow and Google Groups community to get a better understanding of the user's expertise on specific ggplot2 features, referred to as focussed expertise. In the final step the general and focussed expertise approach are combined. In addition we also introduce and discuss our developed tool chain, a collection of programs created to collect, parse, and analyse data from Google Groups, Stack Overflow, and GitHub.

1.2 Outline

First we present the datasets used in this study in Chapter 2. Here we discuss the dataset we used for Stack Overflow and GitHub, explain how the datasets are structured and what information is useful for our study. Furthermore we discuss the developed tools for Google Groups created to collect and parse the required data and the information that is available in the dataset. In Chapter 3 and Chapter 4 we discuss the general and focussed expertise respectively. Both chapters discuss the implementation of the algorithms, their purpose, and the results of our tests. An overview of related work is given in Chapter 5, followed by a discussion of the tool chain created for this study in Chapter 6. Finally we conclude the thesis in Chapter 7, including a discussion on threats to validity.

Chapter 2

Datasets

To find experts in ggplot2 data is required from Stack Overflow (SO), GitHub (GH), and Google Groups (GG). A data dump from the Stack Exchange network, including Stack Overflow, is available at the Internet Archive[1]. The GHTorrent[2] project is used to get all the required Github data. There are, however, no available data dumps for Google Groups. Thus we create our own data dumps by developing a tool to scrape all the required data.

2.1 Google Groups

For the Google Groups community a tool, called 'Google Groups Crawler'[3], was developed because there were no available data dumps. The public available web crawlers for Google Groups at the time of writing were either too limited[4] or outdated[5][6][7] and no longer functional. The developed 'Google Groups Crawler' works by extracting the content of specific html elements using their unique class name, e.g. elements containing title, name, data, and content (questions and answers). Unfortunately the layout of the Google Groups website changes often, once every 1-2 months, causing the class names to change. The source code of the crawler is however structured such that is cost minimal effort to update the class names, making it easy to fix the crawler when it is outdated. As alternative there is a plain-text view for each topic, which does not have dynamic class names. This makes it possible to use XPath queries to directly extract the required data, which gives a significant performance boost. The plain-text view has however some limitations which are discussed below.

```
https://groups.google.com/forum/?_escaped_fragment_=forum/ggplot2[1-100]
```

The link above lists the first 100 topics of the Google Group ggplot2. By using this page as starting point the crawler collects the ids and titles of all topics. In the next step this information is used to collect the posts in each topic, containing information about the post's author, avatar, post date, and the content. On Stack Overflow there is a clear distinction between content, i.e. questions or answers, however Google Groups is more like a mailing list and displays it as a conversation. To distinguish questions and answers on Google Groups we assume the first post is the question and all responses are answers.

```
https://groups.google.com/forum/?_escaped_fragment_=topic/ggplot2/ZbgnsPMCJa8
https://groups.google.com/forum/m/#!topic/ggplot2/ZbgnsPMCJa8
https://groups.google.com/forum/?fromgroups#!topic/ggplot2/ZbgnsPMCJa8
```

[1] https://archive.org/details/stackexchange/
[2] http://ghtorrent.org/
[3] https://github.com/josdemmers
[4] http://rud.is/b/2013/09/25/scraping-content-from-google-groups/
[5] http://saturnboy.com/2010/03/scraping-google-groups/
[6] https://sputnikus.github.io/google_groups_scrape
[7] https://github.com/icy/google-group-crawler

There are three alternatives for collecting post data as listed above. The last part of each URL indicates the Google Group ID (ggplot2) and the topic ID (ZbgnsPMCJa8). The first link is the most basic one, it gives a plain-text view with the exception of some URLs and a simple table layout. The absence of an advanced layout makes it easy to extract the required information. Unfortunately avatars are absent in this layout, which are useful when identifying users. The second and third link look like normal web pages and both contain the avatar of a user. The difference between the two is that the second link is optimized for mobile devices and the third link for normal browsers, and thus uses more advanced ajax/javascript features. The second link is used in the 'Google Groups Crawler' because it contains the same information as the third one but has a simplified layout which makes it easier to extract the required information.

Besides the 'Google Groups Crawler' tool the 'Google Groups Parser' is developed to create the actual data set. For a complete overview of the tool chain see Chapter 6. The crawler is used to save all data in a useful format, i.e. single file per topic containing data of each participant. For further analysis the parser converts those files into a single file for users, topics, and statistics, resulting in the following files: GG_users.xml, GG_topics.xml, and GG_stats.xml. The files contain all the Google Group data from 20/11/08 until 05/02/15, i.e, from the first available post until the latest one available at the time the web crawler was started. The GG_users.xml file contains a list of all users, 2198 to be exact, that participated in the ggplot2 group and contains for each user the ID, Name, Avatar, and Email. Note that the email address is not available for all users because Google Groups protects email addresses by captcha. Instead of solving the captcha's for all users only the top 100 contributors were selected because solving captcha's for all users by hand is a very time consuming task. In addition we are only interested in experts and by limiting ourself to the top 100 all experts are still included. Here we assume that potential experts have made enough contributions in the Google Group to be able to tell whether or not a user is an experts, users not in the top 100 have only 1-2 posts and are by this assumption no experts. The GG_topics.xml file contains a list of all topics with their Id and Title, and posts associated with each topic. For each post the Date, Author, and Content is available. The last file is GG_stats.xml, it contains a user Id and a number of expertise measurements, which will be discussed in Chapter 3.

2.2 Stack Overflow

For the Stack Overflow community a data dump is available at the Internet Archive. The data dump used in this study is from September 2014 and contains all public data since 08/08/2009. Unfortunately since the beginning of 2014 the data dumps are anonymized and no longer contain email hashes. However, by merging the data dump with an older version, March 2013, it was possible to add the email hashes of some users to the current data dump. Only the users who joined Stack Overflow after March 2013 do not have an email hash.

Because of the available data dump it was not necessarily to create a web crawler, only a parser was created to process and filter the data dump. For a complete overview of the tool chain see Chapter 6. With the 'Stack Overflow Parser' files similar to the 'Google Groups Parser' are created, including SO_Posts_filter.xml, SO_Users_filter_hash.xml, and SO_stats.xml. The SO_Posts_filter.xml file contains detailed information of all posts with the ggplot2 tag. The following attributes are of interest:

Id Unique identifier of post.
OwnerUserId Unique identifier of author.
PostTypeId Indicates post type, '1' for question and '2' for answer.
ParentId (Optional) For answers it indicates the unique post Id of the related question.
CreationDate Creation data of post.
Title Title of post.
Body Content of post.
Tags Tags used to categorize post.

The SO_Users_filter_hash.xml file contains detailed information of all users, 3684 to be exact, who

were active in posts with the ggplot2 tag. An email hash is available for 2320 of 3684 users. The following attributes are of interest:

Id Unique identifier of author.
DisplayName User name of author.
EmailHash MD5 hash of email, only available before 2014.
ProfileImageUrl Contains in some cases a gravatar url with a MD5 email hash. Still available in current data dumps.

The last file is SO_stats.xml, it contains a user Id and a number of expertise measurements, which is discussed in Chapter 3.

2.3 GitHub

For GitHub there is a data dump available via the GHTorrent project[12], initial design and implementation by Georgios Gousios. GHTorrent is an effort to create a scalable, queriable, offline mirror of data offered through the Github REST API[8]. GHTorrent recommends to use their MongoDB query service, there is also the possibility to download the raw MongoDB dataset but downloading and restoring the dataset in MongoDB is very time consuming. As alternative there is also a query service and dataset for MySQL, however this query service doesn't contain the content of issues. Therefore the MongoDB query service is used to create a new dataset containing only data from the ggplot2 project. After querying MongoDB the 'GitHub Parser' tool is used to convert the MongoDB results in a useful format. For a complete overview of the tool chain see Chapter 6. The ggplot2 dataset contains data from 02/12/2009 until 24/02/2015, that is, from the first available ggplot2 issue until the latest issue available at the time when requesting the data dump. Although the MongoDB query service is available for the public you need to make a request first to gain access. The GHTorrent MongoDB consists of 16 collections[9], the collections users, issues, and issue_comments are used in this study. After filtering the data for ggplot2 users the collection user contains 291 users, of which 78 have an email address and 277 have a gravatar id. The following attributes in this collection are of interest:

Id Unique identifier of user.
Login Login name.
Name User name.
gravatar_id Gravatar id, MD5 hash of email address.
Email Email address.

The collection issues contains the following attributes that are of interest:

Number Unique identifier of issue.
Title Title of issue.
Body Content of issue.
User.Id Unique identifier of user.

The issue_comments collection contains the following attributes that are of interest:

Id Unique identifier of comment.
Issue_id Unique identifier of issue. Is used to link issue comments to the correct issue.
Body Content of comment.
User.Id Unique identifier of user.

The parsed data is saved in GH_users.xml, GH_posts.xml, and GH_stats.xml. GH_stats.xml is discussed in Chapter 3.

[8] https://developer.github.com/
[9] http://ghtorrent.org/mongo.html

2.4 Merged Communities

Stack Overflow, Google Groups, and Github are merged to get a better understanding of the user's expertise when a user is active in multiple communities. Merging the communities is done in two steps, first Stack Overflow and Google Groups are merged, this result is then merged with GitHub. The order of the merge is not of importance, however merging the communities one by one is. By merging the communities separately we can better monitor the number of matches found in each community.

To be able to merge Stack Overflow and Google Group a MD5 hash is created from the email addresses of Google Group users. When a matching hash is found in Stack Overflow the two users are merged. A similar approach has been used by Vasilescu et al. [37] for combining Stack Overflow and GitHub users. However, only Stack Overflow users registered before March 2013 have an EmailHash attribute that can be used to find a match, for newer users this attribute is no longer available because of privacy concerns. An alternative approach is to use the attribute ProfileImageUrl, there are 787 users who have a ProfileImageUrl with gravatar but only 160 contain an email hash. This makes its use rather limited but by combining this information with the already available MD5 hashes from the EmailHash attribute additional users might be identified. Unfortunately, in the ggplot2 case study the users with an useful ProfileImageUrl attribute already had an equal EmailHash attribute. However if in a future work this case study is expanded to projects other than ggplot2 it could still be useful to identify additional users.

The next step is to merge GitHub users by using the attributes Email and GravatarId. The Email attribute is used to find matching Google Group users, in addition by hashing the Email attribute and comparing it to the attributes EmailHash and ProfileImageUrl matching Stack Overflow users can be found. The GravatarId is a MD5 hash of an email address and can also be compared to the attributes EmailHash and ProfileImageUrl. By hashing the Email attribute of Google Group it can be compared to GravatarId as well.

The resulting user list contains 6022 'unique' users with 151 matches across the communities. It is likely the list contains duplicates because for a real unique list of users the email addresses or hashes of all users are required. Unfortunately they are not always made available because of privacy and spam concerns. Table 2.1 gives an overview of the number of ggplot2 users across multiple communities and the total number of users after merging.

	Users	Email (Hash)	Gravatars
Stack Overflow	3684	2320	160
Google Groups	2198	154	NA
GitHub	291	78	277
Merged	6022		

Table 2.1: Ggplot2 users across multiple communities.

Chapter 3

General Expertise

After collecting the data from the Q&A communities and linking the identities of users active in multiple communities the next step is to find the level of expertise of each user. Defining the expertise of users is done by an iterative process. The first iteration focusses on a general level of expertise, i.e., it is known a user is an expert in ggplot2 but there is no data for what specific features he has any expertise. The next iteration focusses on topic specific expertise by analysing answers posted by users, this allows us to identify experts for specific ggplot2 features. The general expertise will be discussed in this chapter, whereas the focussed expertise will be discussed in Chapter 4.

For the expertise measurements in the first iteration three methods by Zhang et al. [42] are used. Those methods are chosen because the data required by those algorithms can be extracted from the data sets, the algorithms are simple to implement, and the selected algorithms vary from basic to more advanced were each algorithm addresses the limitation of their more basic variant. There are more approaches available to measure a user's level of expertise but those will be discussed in Chapter 5.

3.1 Implementation

The datasets discussed in Chapter 2 each have a file containing statistical information. The file is named stats.xml, with a prefix to distinguish the different communities, i.e., GG_, SO_, and GH_. Besides the user Id to identify users stats.xml contains the following attributes:

Answers The total amount of answers a user has given.
Questions The total amount of questions a user has asked
ZScore Algorithm to calculate an expertise rank using amount of answers and questions, introduced by Zhang et al. [42].
InDegree The total amount of people a user has helped.
OutDegree The total amount of people that helped the user.
ZDegree Algorithm to calculate an expertise rank using the indegree and outdegree of a user, introduced by Zhang et al. [42].
ExpertiseRank Algorithm to calculate an expertise rank based on PageRank[25], introduced by Zhang et al. [42].

3.1.1 ZScore

The ZScore measurement assumes that user attributes Answers (a) and Questions (q) indicate expertise and lack of expertise. By using those two attributes a level of expertise can be calculated.

$$z = \frac{a - q}{\sqrt{a + q}} \tag{3.1}$$

The algorithm works by comparing the number of answers and number of questions of a user. When a user asks more questions than he gives answers it indicates a lack of expertise, resulting in negative ZScore. While a higher number of answers results in a positive ZScore, indicating a higher level of expertise. Users asking and answering an (almost) equal amount of questions will have a ZScore close to 0.

3.1.2 ZDegree

ZDegree is very similar to ZScore as it also uses attributes indicating expertise and a lack of expertise. However, ZScore has a limitation that is addressed by ZDegree. With ZScore the number of answers greatly influences the level of expertise, e.g., when a user helps the same user multiple times with similar questions, or when a conversation occurs between the asker and helper to clarify the question. This will increase the level of expertise of the user answering the questions, although in practice answering such questions does not indicate a higher level of expertise. Note that *helps* in this context equals to answering a users question. ZDegree addresses this false increase of expertise by using the amount of people one helps instead. For Q&A communities and any other social network this can be calculated using the indegree (in) and outdegree (out) of users.

$$z = \frac{in - out}{\sqrt{in + out}} \tag{3.2}$$

The indegree is measured by counting the total amount of people a user has helped and outdegree by counting the total amount of people that helped the user.

3.1.3 ExpertiseRank

The ExpertiseRank algorithm also makes use of the indegree and outdegree of users. In addition it addresses a limitation of the ZDegree algorithm. Both algorithms look at the amount of people one helps but ExpertiseRank also takes into account the expertise of other users, i.e., when helping a user with a high expertise level it should increase the level of expertise more than when helping someone with a low expertise level.

$$ER(A) = (1 - d) + d(ER(U_1)/C(U_1) + ... + ER(U_n)/C(U_n)) \tag{3.3}$$

ER(A) ExpertiseRank of user A.
ER(U_i) ExpertiseRank of users helped by A.
d Damping factor, value between 0 and 1.
C(U_i) The total number of users helping U_i.

The damping factor has two uses, first of all it gives a minimum ExpertiseRank in case a user has not helped a single person. In addition it reduces the extend of which the ExpertiseRank of a user influences the ExpertiseRank of another user. The ExpertiseRank is calculated by taking the sum of the ExpertiseRank of users U_i divided by $C(U_i)$ for $0 < i \leq n$, where n is the indegree of A. The resulting sum is multiplied by the damping factor and finally $(1 - d)$ is added resulting in the ExpertiseRank of A.

Comparing this to PageRank the damping factor simulates the chance a user keeps clicking links, when stopped the user will jump to another page at random, this is the so called 'Random

surfer model'. Users U_i are equal to all sites that link to A and $C(U_i)$ is the total amount of outgoing links on page U_i.

The ExpertiseRank algorithm is implemented in C# and a code snippet is available in Appendix D. Although Zhang et al. [42] described the algorithm in their paper the source code itself was not available. The ExpertiseRank algorithm in this study is therefore based on a PageRank implementation[1]. Important to mention for the implementation is that the PageRank algorithm is a probability distribution, for ExpertiseRank this means the result indicates the probability that a person who asks a random question is helped by a particular user. The implementation works by calculating $ER(A)$ using an iterative approach.

The ExpertiseRank starts at an initial value of 1. The exact initial value is of no importance, the end result will be the same. The only difference is the amount of iterations it takes to reach a steady state. Each iteration adjust the ExpertiseRank values until a steady state is reached, this is to prevent a infinite number of iterations. When it reaches a steady state it means additional iterations have little to no effect. In our implementation a steady state is reached when the change between iterations is smaller than ϵ, for $\epsilon = 0.00001$.

To give a small example the ExpertiseRank of user C, as shown in Figure 3.1, is calculated using Equation 3.4.

$$ER(C) = (1 - d) + d(ER(A)/2) + ER(B)) \tag{3.4}$$

With this equation we calculate the ExpertiseRank for user C. The first part, $(1 - d)$, gives user C a minimum expertise value. The second part is based on the ExpertiseRank of all users helped by user C. That is, it takes the sum of those expertise values divided by the total amount of people who helped each user. In this example user C has helped A and B, as shown by the incoming arrows of C in Figure 3.1. User A is helped by B and C, as shown by the outgoing arrows of A, resulting in $ER(A)/2)$. User B is only helped by C, as shown by the outgoing arrows of B, resulting in $ER(B)/1$. Combining those as shown in the equation results in the ExpertiseRank of user C.

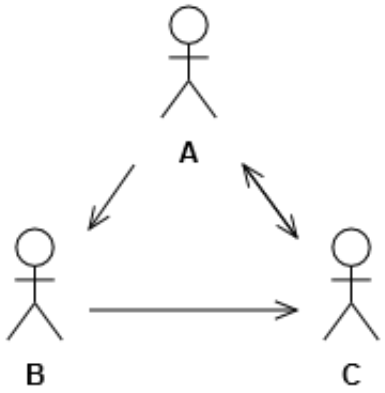


Figure 3.1: ExpertiseRank example.

3.2 Validation

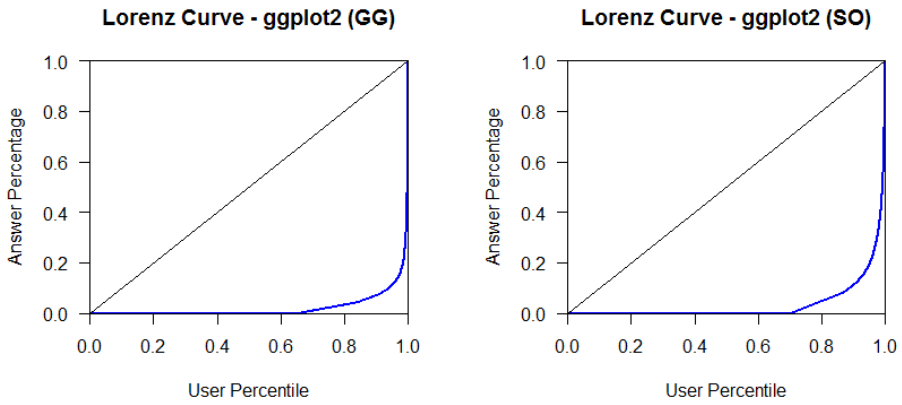
The analysis and validating part is partly done by tools, the automated part, and there is a manual validation, by creating a survey.

[1] https://github.com/mmallett/cpre310-pagerank

3.2.1 Automated

The data required for each of the algorithms discussed in Sections 3.1.1, 3.1.2, and 3.1.3 differ slightly although all data is available in the data set. Another difference is the complexity of each algorithm. In this section the results of the algorithms will be discussed to be able to tell whether or not more advanced algorithms have an advantage for this study, i.e., do more advanced algorithms give a better understanding of users expertise.

When identifying experts in a community it is not the whole user base that is of interest but a smaller part of it, that is, the top contributors of the community. As shown in Figure 3.2 only a small percentage of users is responsible for most answers given in the ggplot2 community.



(a) Google Groups

(b) Stack Overflow

Figure 3.2: Lorenz curve of the ggplot2 community at Google Groups and Stack Overflow.

The Lorenz curve, developed by M.O. Lorenz in 1905 [20], was originally used for representing inequality of the wealth distribution. In Figure 3.2 it is used to represent the inequality of the amount of answers given by users. The vertical axis shows the cumulative percentage of answers and the horizontal axis the percentile of users. Furthermore the diagonal represents equality, i.e., all users give an equal amount of answers. The actual inequality in the Google Groups community, Figure 3.2a, is represented by the curved line, how bigger the area between the diagonal and the curved line the greater the inequality. In this case 1% of the Google Group users posted 80% of all answers.

When comparing the top 20 contributors, Appendix Table C.1, there is small difference in order between the algorithms. However, besides a few exceptions the top 20 of each algorithm contains the same users. To analyse this in more detail a scatter plot is created, Figure 3.3, to verify whether or not there is a high correlation between the algorithms. Notice that for this analysis Google Groups is used, additional scatter plots for Stack Overflow are available in Appendix C.3. Figure 3.3a contains scatter plots for only the top 50 contributors while Figure 3.3b contains scatter plots for all contributors. Table C.1 already indicated that each algorithm had almost identical top 20 lists, this is verified by Figure 3.3a. The scatter plot between the ZScore and ZDegree start to diverge a little for lower rankings but overall they remain very similar. This is expected because both algorithms are very similar. However, scatter plots between the ExpertiseRank and either ZScore or ZDegree diverge stronger for ranks beyond the top 20. Although we notice the diverging plots in the top 50 the algorithms remain strongly correlated when looking at the overall picture, Figure 3.3b. We notice however odd horizontal and vertical lines, especially in scatter plots for the ExpertiseRank. Upon closer inspection those lines are caused when of the two algorithms of the scatter plot one algorithm gives multiple users the same expertise value while the other algorithm is able to give unique expertise values. Since ZScore and ZDegree are having difficulties to distinguish lower ranked users and ExpertiseRank handles this better the vertical and horizontal

lines are more noticeable by the ExpertiseRank scatter plots. In addition to the scatter plots R is also used to calculate the correlation, available in Appendix E.1. Different correlation methods are Pearson, Kendall, and Spearman for which both Kendall and Spearman are suitable for ranking data. The Spearman method is used in our calculations, which is also the default method used by R. Similarly to the Google Groups scatter plots, Table 3.1 indicates a very high correlation between the ZScore and ZDegree and a slightly lower correlation when comparing ExpertiseRank. Stack Overflow shows however a higher correlation between the ExpertiseRank and ZDegree, which is also noticable by Figure C.2.

	ZScore	ZDegree	ExpertiseRank
ZScore	1.00	0.94	0.82
ZDegree	0.94	1.00	0.77
ExpertiseRank	0.82	0.77	1.00

(a) Google Groups

	ZScore	ZDegree	ExpertiseRank
ZScore	1.00	0.86	0.82
ZDegree	0.86	1.00	0.93
ExpertiseRank	0.82	0.93	1.00

(b) Stack Overflow

Table 3.1: Correlation between algorithm rankings.

Taking the similarity and the complexity of the algorithms into account the ZDegree algorithm will be used in the final implementation of the tool chain, a complete overview of the tool chain is available in Chapter 6. The ExpertiseRank would be a better choice when ranking the whole user base, but since we are only interested in the top contributors ExpertiseRank does not have any advantages. The reason ZDegree is used instead of ZScore is because ZDegree counts the users that were helped multiple times by the same person only once. The advantage of this is over ZScore is that multiple responses on the same question do not result in a higher ranking as mentioned in Section 3.1.2.

In addition to comparing the algorithms it is important to know they actually give realistic results. To verify this a survey is created for the ggplot2 community at Google Groups, which will be discussed in Section 3.2.2.

3.2.2 Survey

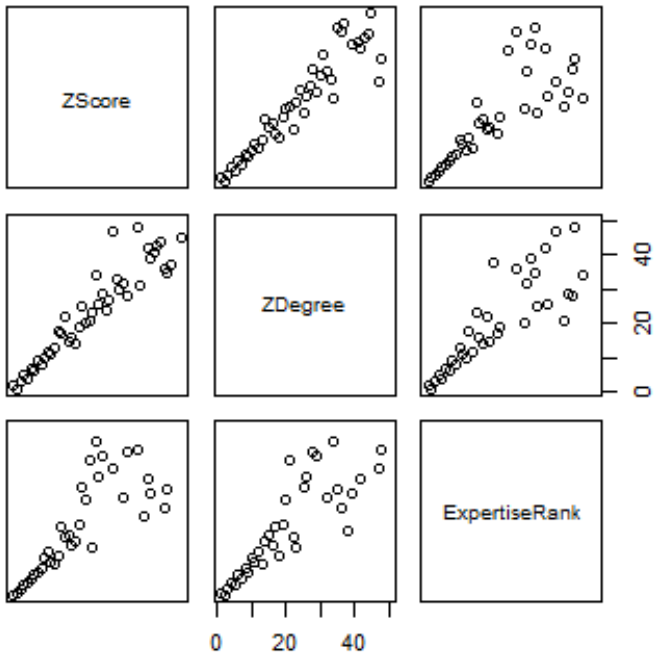
The goal of this survey is to answer the following research questions:

RQ1: Do automated expertise algorithms work well enough compared to manual ratings?

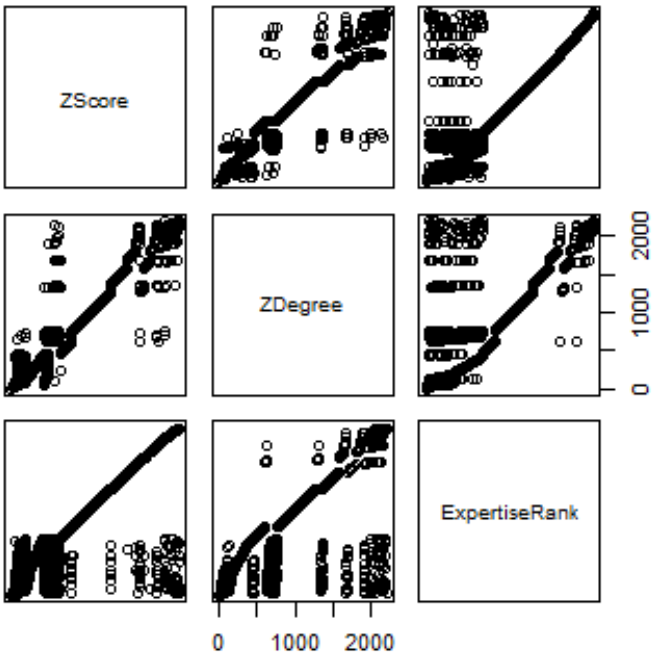
RQ2: What expertise algorithm has the best performance, i.e. has the best match with the manual ratings. Or are the differences so insignificant that it does not matter what algorithm you use?

By asking ggplot2 users to rank some of their fellow ggplot2 users we expect the survey to give more insight into the expertise of users. Comparing those manual ratings with the algorithm rankings will allow us to test the quality of the algorithms. The main purpose is to verify the top ranked users from the algorithms, as those are the users we are interested in. Furthermore the lower ranked users often have only 1-2 posts which makes their algorithm ranking unreliable. Because of those unreliable results and the fact that we are only interested in the top ranked users the verification of the results focusses on the top ranked users. It will probably be difficult to use the survey to verify the exact order of users but it should not be a problem to make a distinction between expert users and regular users.

The survey starts with a short introduction explaining the purpose of this survey, followed by



(a) Top 50 contributors



(b) All contributors

Figure 3.3: Scatter plots showing the similarities in ranking between the different algorithms (Google Groups).

the instructions. The message, as posted on Google Groups[2] and included in the survey, is shown below.

> Dear ggplot2 users,
>
> As part of my Master thesis I'm trying to automatically identify experts in Q&A communities, for which I use ggplot2 as case study.
> To validate the results which I have so far, I have created this survey about 11 random ggplot2 users. Those users were selected from a list of users ranked by expertise by a number of different algorithms. We expect that answering the survey will take not more than 15 minutes of your time. If you are interested in the results of this study you can provide your email address at the end of this survey.
>
> This thesis is done at the Eindhoven University of Technology supervised by Alexander Serebrenik. No commercial party is involved in this study. The responses will be used in an anonymized form for the master thesis and eventually a scientific paper. The response will not be shared with any third party, commercial or other.
>
> The survey is anonymous.
>
> Jos Demmers
> Master student Computer Science Engineering
>
> #Instructions:
> Each ggplot2 user listed below can be ranked from 1-10, where 1 stands for someone who just started learning / using ggplot2 and 10 for someone who has mastered ggplot2 completely. Please rate the user accordingly. When you are not familiar with one of the users below you can click on the provided link to view a list of recently made posts by that user.

Figure 3.4 shows an example of a survey question. The questions are constructed as follow: They start with a name and link to a list of the most recent post by that user. This list can be used by participants when they are not familiar with a user. Followed by a rating from 1-10 to rate expertise of users and finally an additional comment field if a participant wants to enter some additional remarks. Furthermore, besides the questions the survey also contains two additional email fields, allowing participants to specify whether or not they are interested in a follow up survey and/or the end result of this study.

The users in the survey are selected using a method similar to stratified sampling, the difference being that some users are excluded. The users are randomly selected from two groups from the ZDegree ranking, the top 20 and the top 100+. Eight users are selected from the top 20 and three from the top 100+. With the selection of the top 20 users it can be verified whether or not the users participating in survey view them as experts as well. The selection of top 100+ users has two purposes. First of all to verify if the algorithm has made the correct distinction between expert users and regular users. Secondly as a control group to check whether or not a participants ratings are valid, i.e. for everyone filling in the survey seriously it is unlikely they will rate top 100+ equal as top 20 users.

The survey was open for six weeks, it started late April and closed at the end of May. The response rate of the survey was rather low. Although the ggplot2 community at Google Groups has almost 2200 unique users only 7 users responded. However, looking at Figure 3.2a it is not unexpected, only a small number of users are responsible for the majority of the posts. The remaining users often have only a handful of posts, as visualized by Figure 3.5. It shows that the post count of most users is below 10. As a final remark, we also looked into the amount of frequent posters, i.e. users ranked in the ZDegree top 50, that were active in recent months. From this top

[2] https://groups.google.com/forum/#!topic/ggplot2/ErMj24nL4vo

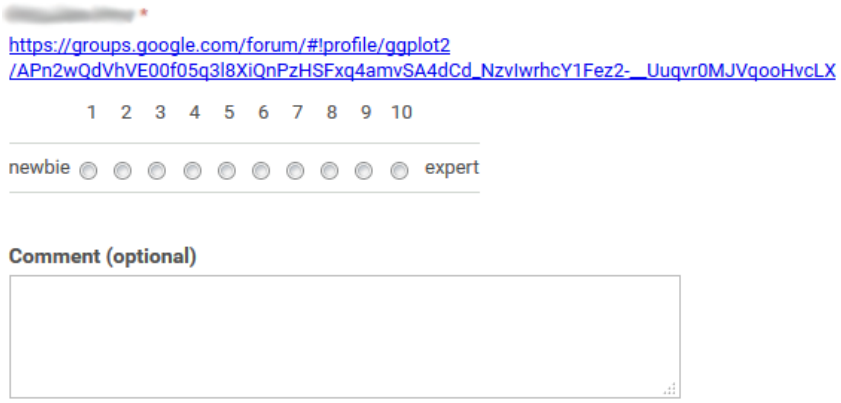


Figure 3.4: Survey example question.

50 only 20% was active in the last 2 months. Although the survey was anonymous it is likely the 7 participants of the survey are users from this active group. This also explains the low response rate, as the frequent posters are most likely to fill in the survey.

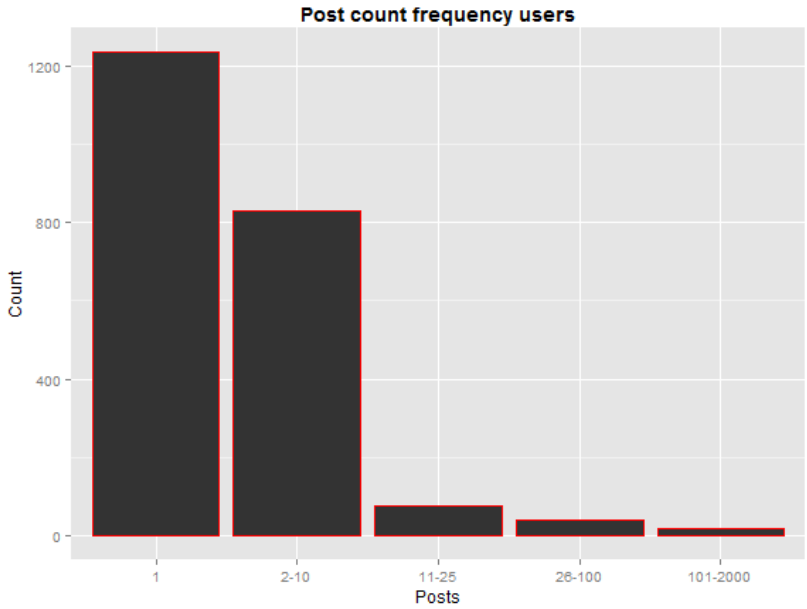


Figure 3.5: Post count frequency Google Groups users. Horizontal axis shows five different categories of users grouped by the amount of posts they made.

In Figure 3.6 the ratings are shown for each user. Users a-h are seen as experts by the algorithms and users i-k as regular users. The horizontal axis shows the ratings from 1-10 and the vertical axis the frequency a user is given a specific rating. All results are anonymized to protect the privacy of participants and ggplot2 subscribers, a user mapping is available in Appendix A. The names and recent posts of each user were however available in the survey, this was needed so that participants of the survey knew who they were rating and were able to check the quality of the posts.

Table 3.2 shows the mean and median ratings for each user in the survey. When inspecting the average ratings the top 20 and the top 100+ users (last three rows) are clearly distinguishable, this

matches with the algorithms. To be more precise, the top 20 users have ZDegree values varying from 21.38 to 5.52, with and average of 10.63 and the values of top 100+ users vary between 1.41 and −6.08, with an average of −0.90. Making them like the survey results clearly distinguishable. In addition to distinguish experts and regular users the top 100+ is also used as control group, because of the distinguishable ratings it can fulfil this role. Looking back at the individual ratings, Figure 3.6, there are two exceptional ratings for the top 100+ users. One participant rated two users 8 out of 10 while all other participants gave ratings varying from 1 to 5. Inspecting the contribution and activity of those two users did not yield any argument for such a high rating, however the participant's other ratings are in line with the other participants. Therefore we saw the ratings as valid and kept the survey response. In addition to verify that participants see the top 20 users as more knowledgeable we also compared how the users were ranked by rating. Although the users ranked in the top 5 by the algorithm are rated between 8-10 and among the top of the user ratings there are little similarities between the remaining users. Furthermore there is also some variance between participants ratings for each user, making it not very reliable for verifying the algorithm rankings. Therefore the survey is only used to verify the distinction between expert and regular users.

ID	Average	Median	ZDegree Rank	ZDegree value
474	8.86	9	1	21.38
270	8.57	8	3	17.38
195	9.43	10	5	13.91
209	7.29	7	8	11.04
1703	7.71	8	10	8.60
898	6.00	7	11	8.55
858	6.14	6	14	6.93
1025	7.86	8	18	6.41
2092	3.14	3	1570	-1.41
1956	3.57	3	1680	-1.63
1449	3.57	3	2187	-4.08

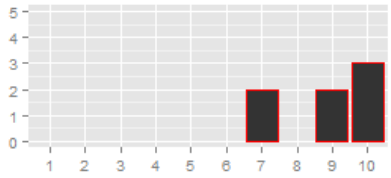
Table 3.2: Average expertise ratings from survey, sorted by ZDegree.

Overall Agreement

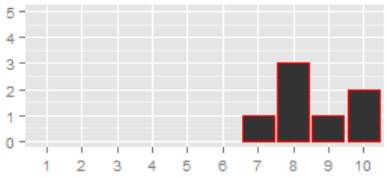
To measure the overall agreement between the participants of the survey we calculate the Intraclass Correlation (ICC) [33]. ICC can be used to describe how strongly units in the same group resemble each other. In this context the units represent the ratings made by the participants (judges) and the different groups represent each of the users (targets) they had to rate. The three base forms of ICC are:

ICC1 Each target is rated by a different judge and the judges are selected at random.
ICC2 A random sample of k judges rate each target.
ICC3 A fixed set of k judges rate each target.

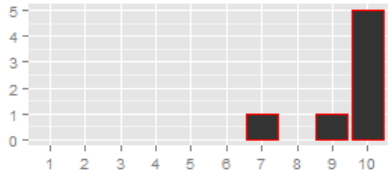
We used the ICC3 variant as we have a set of 7 participants (judges) who each rated all users (targets). The ICC varies between 0 and 1, where 0 means no agreement and 1 total agreement. The resulting ICC value for our ratings was 0.68. This value is interpreted by some researchers as a fair agreement [39], indicating that there is at least some agreement among the participants. This is also noticeable in Figure 3.6 as the agreement varies per user. For the ratings for users a-c in Figure 3.6 we see the most agreement, whereas the ratings for users f,g,j,k show little agreement. Furthermore for the overall agreement all participants agree with the difference between regular and expert users.



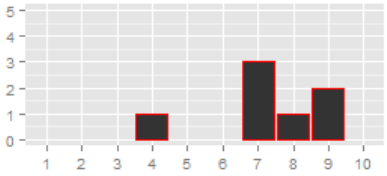
(a) User id 474



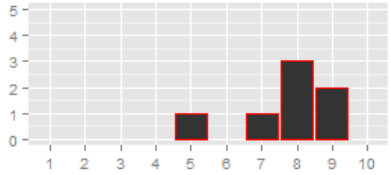
(b) User id 270



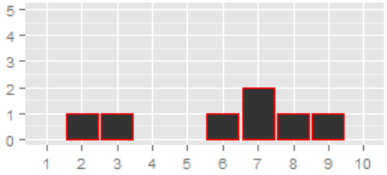
(c) User id 195



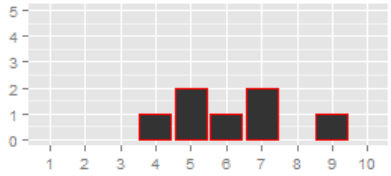
(d) User id 209



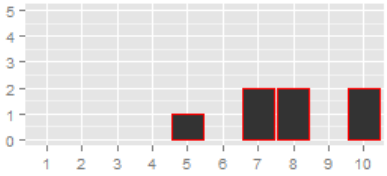
(e) User id 1703



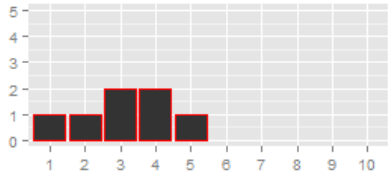
(f) User id 898



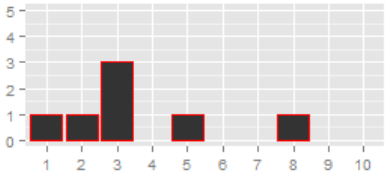
(g) User id 858



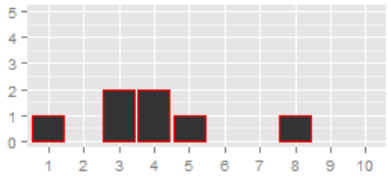
(h) User id 1025



(i) User id 2092



(j) User id 1956



(k) User id 1449

Figure 3.6: Expertise ratings from survey.

3.3 Conclusion

The general expertise is implemented to define an overall level of expertise for users, to be able to recognize experts in a community. In addition we wanted to achieve two things. First of all we want to make a comparison of expertise ranking algorithms to be able to test if more advanced algorithms have an advantage for this study. Secondly, to verify whether or not the rankings by the expertise algorithms are comparable to manual ratings.

The expertise rankings of the used algorithms, ZScore, ZDegree, and ExpertiseRank were all very similar as indicated by both the correlation (Table 3.1) and the visualization of the rankings using scatter plots (Figure 3.3). The advantage of the more complex ExpertiseRank over the other algorithms is that it is better in making distinctions between the lower ranked users. However, as we are interested in the experts of the community this does not have any advantage for us. Our final algorithm of choice is the ZDegree. With regards to complexity it is similar to the ZSCore but has one small improvement, i.e. answering the same user multiple times will not increase your ZDegree.

Our findings from the survey (Table 3.2) indicated that the manual ratings given by the survey participants are comparable to the algorithm rankings. Although they are not precise enough to verify the exact ranking of expert users they can successfully be used to make a distinction between experts and regular users.

In Chapter 4 we will extend the general expertise by describing the expertise of users on specific subjects related to ggplot2, the so called focussed expertise.

Chapter 4

Focussed Expertise

The general expertise discussed in Chapter 3 allows us to identify experts within the ggplot2 community and is the first step in the iterative approach to describe the expertise of users. The second and final step in this iterative process is to be able to describe the expertise of users on specific subjects related to ggplot2 features, i.e., the focussed expertise of users. The focussed expertise is not a separate approach to measure users expertise but is an extension on the general expertise. The focussed expertise is an extension in the sense that the general expertise is only able to identify a user as an expert but can not specify the ggplot2 features a user is familiar with. By combining the two approaches we can tell for each expert the ggplot2 features they are knowledgeable about.

4.1 Implementation

4.1.1 Topic Analysis

To be able to specify the terms users are familiar with we have to analyse their activity in the community. In Q&A communities this can be done by analysing users' posts. Each user's post is checked for important keywords that describe the field of expertise of a user. The keywords were selected by two different approaches: TF-IDF and a custom word list based on the documentation of ggplot2, which we refer to as 'Custom' approach. Each approach has its advantages and disadvantages.

TF-IDF

The idea behind the use of TF-IDF is to have a general approach that is able to automatically select words of importance from a collection of documents (posts). Note that 'words of importance' in this context is how TF-IDF interpreted a word, this approach does not allow to select words yourself. In addition TF-IDF works independently of the contents' subject. Meaning, not only for ggplot2 related content but for any project of interest. This approach has the advantage that it can be used for other expertise studies as well without making any significant changes.

TF-IDF stands for Term Frequency - Inverse Document Frequency and is able to give weights to terms in a document based on their importance. The weight of a term depends on the frequency it appears in a specific document and the total amount it appears in the complete collection of documents being analysed. TF-IDF is often used in text mining studies and useful for query like systems for searching through a collection of documents [10][30][32][40]. TF-IDF consist of two parts, Term Frequency and Inverse Document Frequency. Term Frequency is used to measure how frequently a term occurs in a document, in addition Term Frequency is often normalized to take into account documents of different sizes.

$$\begin{aligned} TF(t) = (&Frequency\ of\ term\ t\ appearing\ in\ a\ document) \\ &/\ (Total\ number\ of\ terms\ in\ the\ document) \end{aligned} \tag{4.1}$$

The Inverse Document Frequency measures the importance of a term for a document in relation to the complete collection of documents. For example a term that appears multiple times in a document but is less common in other documents has a high IDF value. Such a term is important because it is a good identifier for the document, e.g., a search query with this term will rank the document high when sorted by relevance. Terms with low IDF values are those who appear frequently in all documents, the so called stopwords.

$$IDF(t) = Log((Total\ number\ of\ documents) \\ / (Total\ number\ of\ documents\ with\ term\ t)) \tag{4.2}$$

TF-IDF is implemented using the text mining (tm) package for R [24]. The R script is available in Appendix E.2. The first step is to convert the collection of documents, i.e., questions and answers posted in the ggplot2 community, to a corpus. After converting the collection multiple transformations can be applied on the corpus using the *tm_map* function. The purpose of the transformations is to reduce the size of the corpus, which can then be used by TF-IDF to process the documents. The following transformations are applied:

- Convert all words to lower case.
- Remove stopwords.
- Strip whitespace.
- Number of regular expression transformations.

The regular expression transformations are used to remove nonsense words by removing all non alphabetic/numeric strings, doubles, punctuation, and strings consisting of numbers followed by letters. Another transformation that is often used in text mining is word stemming, its purpose is to reduce words to their base or root form. However, after applying it on our corpus we noticed some unwanted behaviour on code snippets. While the transformation correctly finds the stem of a word it makes the function names unrecognisable in further analysis and we are therefore not using word stemming. Typical examples are that function set.seed becomes setse, aes becomes ae, and library becomes librari. After applying all transformations the final step is to create a DocumentTermMatrix which results in a list of terms for each document with their TF-IDF values. Our approach using TF-IDF is thus rather naive, we use TF-IDF as offered by the R-framework without making significant changes for the ggplot2 community. Disabling stemming is one of the small changes that was possible with R to make improvements. It was also our intention to have one general approach that could be used for any community and analyse how it performs. On the contrary, our second approach is for ggplot2 specifically.

Custom Wordlist

The Custom approach is specially created for ggplot2 and cannot directly be used for other projects. However, the advantage of such a list is that all words are relevant for ggplot2 and the usage of such words in a document indicates the post was about a specific ggplot2 feature.

The custom word list is created using a R script, Appendix E.3, that extracts all function names from the ggplot2 documentation. No extra transformations are applied on the list of words because in contrast to the TF-IDF approach this does not contain any stopwords, nonsense words, or strings consisting of non alphabetic characters.

4.1.2 Expertise Vector

The expertise vector is a vector used to indicate the terms a user is familiar with and is created for both the TF-IDF and Custom approach. The expertise vectors for all users are created using the Topic Analyser tool, Figure 6.7, which is discussed in detail in Chapter 6.

The terms found in Section 4.1.1 are used as the base for the expertise vector of users and define the dimension of the vector. For the TF-IDF approach there are a total of 39003 terms for

Google Groups and 48444 for Stack Overflow, resulting in a 39003 and 48444 dimensional vector. For the Custom approach there are 176 terms, resulting in a 176 dimensional vector.

The next step is to set the value for each dimension, this is slightly different for each approach. For TF-IDF the DocumentTermMatrix created in Section 4.1.1 is used, which was exported in csv format by R. The DocumentTermMatrix basically contains a list of vectors that are similar to the expertise vectors only they describe a single document (post) instead of a user's expertise. To create an expertise vector for a specific user we select all document vectors that are related to that user, i.e., the questions and answers posted by that user. Combining the selected document vectors results in the expertise vector, for which we have two implementations:

- With term frequency.
- Without term frequency.

The first implementation combines the document vectors by taking the sum of each vector. Resulting in a vector that takes into account the frequency of used terms across all documents. An alternative would be to take the maximum value for each dimension of the vectors. Although using the maximum value gives a good indication of the terms a user is familiar with, the total sum is better to show the frequency of each term. In the second implementation we dismiss the term frequency completely. Instead of taking the sum of the combined vectors we cap the value of each vector dimension at 1. The expertise vectors for the Custom approach are created by checking each user's post for the terms listed in the custom wordlist. When a term is used by a user its corresponding dimension of the expertise vector is set to 1. Term frequency is not used in this approach, the Custom approach focusses only on whether or not a user is familiar with a term not how often the term is used.

4.1.3 Cosine Similarity

The final step is to find matches between a user's expertise and the content of questions and issues. This is realised by measuring the cosine similarity between the expertise vector and the document vector, a measurement often used in text mining for comparing textual information [2][41]. Important to mention is that document vectors in our approach are binary vectors, i.e., each vector dimension's value is either 0 or 1 depending on whether or not the term is present in the document.

The cosine similarity is a measurement between two vectors that compares them by calculating the cosine of the angle between the vectors. A cosine can vary between -1 and 1 depending on the similarity of the vectors. For example see vector A and B, defined in Equation 4.3. Those vectors are exactly the same, have an angle of 0°, resulting in a cosine similarity of 1, i.e., the terms of both vectors are an exact match. Vectors C and D on the other hand are orthogonal, have an angle of 90°, resulting in a cosine similarity of 0, i.e., the vectors do not have a single term in common. Although cosine can vary between -1 and 1 the cosine similarity will vary between 0 and 1 because a term can not appear a negative number of times in a document.

$$A = \begin{bmatrix}1\\1\end{bmatrix}, B = \begin{bmatrix}1\\1\end{bmatrix}, C = \begin{bmatrix}1\\0\end{bmatrix}, D = \begin{bmatrix}0\\1\end{bmatrix} \tag{4.3}$$

For the implementation of the cosine similarity in the Topic Analyser tool, Figure 6.7, three different approaches are implemented and analysed:

- With term frequency.
- Without term frequency.
- Without irrelevant terms.

The approaches will be discussed one by one. For TF-IDF we use all three and for the Custom approach only two, excluding the term frequency. The term frequency approach for TF-IDF is

similar to the regular approach seen in other works on text mining, i.e., two expertise vectors are created from textual data using TF-IDF and are compared by using cosine similarity without any extra modifications to the expertise vectors. The other two approaches are modifications on the regular approach to make it more suitable for expertise matching. The idea behind the approach without term frequency is to get more realistic results regarding the expertise of users. Assume vectors Doc, $User1$, and $User2$ as defined in Equation 4.4.

$$Doc = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, User1 = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 3 \end{bmatrix}, User2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.4}$$

Assuming term frequency indicates more familiarity with a term then $User1$ is more experienced than $User2$ as for both terms the term frequency is higher by $User1$. However, when calculating the cosine similarity, the expertise matching between Doc and $User2$ is better because the vectors are exactly the same even though $User1$ is the more experienced user. By ignoring the term frequency in this approach the expertise matching is equal for both users, preventing that less experienced users are ranked above the more experienced users. In the third and final approach we ignore, in addition to the term frequency, also terms in the user expertise vector that are not relevant for the document. Meaning all terms that do not appear in the document, this is realised by setting the values of each term not present in the document to 0. The reason behind this is shown by the following example. Assume the document vector Doc, defined in Equation 4.5, and the three expertise vectors $User1$, $User2$, and $User3$. When calculating the cosine similarity between Doc and $User1$, and Doc and $User2$ the result is exactly the same. This is expected when looking at it from a mathematical point of view, however it is not the preferred result when matching the expertise of users to documents. $User1$ is as indicated by the vector familiar with all terms used in the document while $User2$ is only familiar with one term, therefore from an expertise point of view $User1$ should have a better match with the document than $User2$. To improve this all irrelevant dimensions of the expertise vector are set to 0 when calculating the cosine similarity, resulting in the vector $User3$.

$$Doc = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, User1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, User2 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, User3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.5}$$

Compared to the regular approach with term frequencies used for TF-IDF we expect the approaches without term frequencies to be able to make a better distinction between the expertise of users because those are more focussed on the actual expertise of users whereas the regular approach is about the textual similarities.

4.2 Validation

To evaluate the results of the expertise matching we use an automated method, implemented in the Topic Analyser tool, and a manual method, by creating a survey.

4.2.1 Automated

The automated validation focusses on questions that have been answered. In Google Groups 559 out of 4866 questions are unanswered and for Stack Overflow 593 out of 6156. To be able to verify the expertise ranking of users the following method is used. First of all the data sets are partitioned into 10 equal sized subsets. One subset is then used as testing set and the remaining subsets as training set. This is repeated 9 times such that each subset is used as a testing set once. By using this 10-fold cross validation the whole data set can be used in the validation process while keeping the testing and training sets separated. The training set is used to specify the expertise vectors

of users, for each question in the testing set we then create an expertise ranking by calculating the cosine similarity between the document vector (question) and the expertise vectors of users. Thus for each question users are ranked by the results of the cosine similarity. Then we check the ranking of all users that answered a specific question and calculate the median ranking. By doing this for all questions in each of the testing sets we are able to evaluate the overall performance of the expertise rankings. Note that with this method we are not able to verify users that have the required expertise but happen not to have answered the question. This part is verified in Section 4.2.2, using a survey.

Alternative Approaches

The comparisons between the different approaches as discussed in Section 4.1 are shown in Table 4.1 and Table 4.2 for TF-IDF and Custom respectively and are tested on the Google Groups data set. The tables show the median results of the 10-fold cross validation. Although this does not allow us to analyse each test separately it gives us a general idea about the overall quality of each of the approaches and is used to decide what approach will be used in the final implementation. For the final selected approach visualisations are created to be able to make a distinction between each test.

	Term frequency	No term frequency	No irrelevant terms
Rank 1-10	207	63	3
Rank 11-20	110.5	82.5	0.5
Rank 21-30	52	101	0.5
Rank 31-40	16	98	0
Rank 41-50	4	51.5	0
Total analysed	391 (91%)	391 (91%)	5 (1%)
Total	431	431	431

Table 4.1: TF-IDF expertise ranking results (GG).

	Term frequency	No term frequency	No irrelevant terms
Rank 1-10	NA	23	216.5
Rank 11-20	NA	55.5	44
Rank 21-30	NA	81.5	13
Rank 31-40	NA	95	4.5
Rank 41-50	NA	33	0
Total analysed	NA	287 (67%)	281 (66%)
Total	NA	431	431

Table 4.2: Custom expertise ranking results (GG).

The rankings in the table are based on the cosine similarity of the document vector (question) and the expertise vectors of users. The remaining columns indicate for each approach the median amount of questions (of the 10-fold cross validation) with a specific median rank of the participating users. The bottom rows show the total amount of questions, for which 'Total analysed' indicates the median amount of questions each approach was able to analyse and 'Total' indicates the total amount of questions that were available in a single testing set of the 10-fold cross validation. Note that the results for term frequency by the Custom approach are empty because the Custom approach does not use frequency data. Analysing the results of TF-IDF in Table 4.1 indicates that using term frequency apparently holds the best results. However, after closer inspecting this is more of a coincidence than a realistic view on the quality of the expertise matching performance. This becomes clear after analysing the results of 'No term frequency' and 'No irrelevant terms'. The cosine similarity used to rank users expertise does not work well with term frequency as explained in Section 4.1.3. The two main reasons are:

- Users with a high term frequency are ranked lower than users with a low term frequency. As explained using Equation (4.4).
- Users with additional expertise that is not relevant for the document are ranked lower than users with no additional expertise. As explained using Equation (4.5).

By solving those issues one by one, respectively by removing term frequency and irrelevant terms the approach gives a ranking that better matches the actual expertise of users. However, as shown in Table 4.1 the results are not very promising. This indicates that the terms selected by TF-IDF are currently not usable to match the expertise of users to documents. Additional transformations and filter techniques are needed to improve the current list of term. The Custom approach, Table 4.2, is more promising. The results indicate that removing the term frequency and irrelevant terms has a positive effect on finding matches between the expertise of users and documents.

Final Approach

From the discussed approaches we select TF-IDF (without term frequency) and Custom (without irrelevant terms) to further analyse the ggplot2 datasets of Google Groups and Stack Overflow. The results in Table 4.3 show the median results of the 10-fold cross validation. A visualisation of the ranking results is shown in Figure 4.1, which makes a distinction between each of the tests.

	TF-IDF (GG)	Custom (GG)	TF-IDF (SO)	Custom (SO)
Rank 1-10	63	216.5	94	145.5
Rank 11-20	82.5	44	78.5	83
Rank 21-30	101	13	82.5	46
Rank 31-40	98	4.5	69	24
Rank 41-50	51.5	0	40.5	7
Total analysed	391 (91%)	281 (65%)	362.5 (65%)	304.5 (55%)
Total	431	431	556	556

Table 4.3: Expertise ranking results.

Table 4.3 shows the results for the TF-IDF and custom approach and contains the ranking results for both Google Groups and Stack Overflow. The ranking is based on the cosine similarity of the document vector (question) and the expertise vectors of users. For each ranking the table indicates the median amount of questions (of the 10-fold cross validation) where the participants had that specific median rank. The bottom rows show the total amount of questions, for which 'Total analysed' indicates the median amount of questions each approach was able to analyse and 'Total' indicates the total amount of questions that were available in a single testing set of the 10-fold cross validation.

Noticeable in Table 4.3 is that for both approaches not all documents (questions) are analysed. The amount of questions that could be analysed is further reduced by splitting the data in training and testing sets. Therefore for practical purposes an analysis is also done using a complete data set for training, available in Appendix C.4. Three reasons why questions could not be analysed are:

- There is no expert user with a matching expertise for the question.
- 'Empty' document vector, i.e. all values are set to zero. This happens when a document does not contain a single known term.
- Question answered by user who is not recognized as an expert.

The first two reasons are closely related, for all but one case an 'Empty' document vector is the cause no matching expert could be found. This problem occurs mostly in the Custom approach and seldom in the TF-IDF approach. For the Google Groups data set TF-IDF could not analyse

0.1% of the questions because of this and the Custom approach 31%. For Stack Overflow there were no 'empty' document vectors when using TF-IDF, however for the Custom approach 16% could not be analysed. In addition the Custom approach for Stack Overflow had 1 question that could not be analysed because there was no matching expert even though the document vector was not empty. The reason 'empty' document vectors mostly occur by the Custom approach is because of the used word list. This problem indicates that although the current list of terms can describe a users expertise improvements could be made by expanding the list further. The final reason is when questions are answered by users who are not recognized as expert, this explains the remaining questions that could not be analysed.

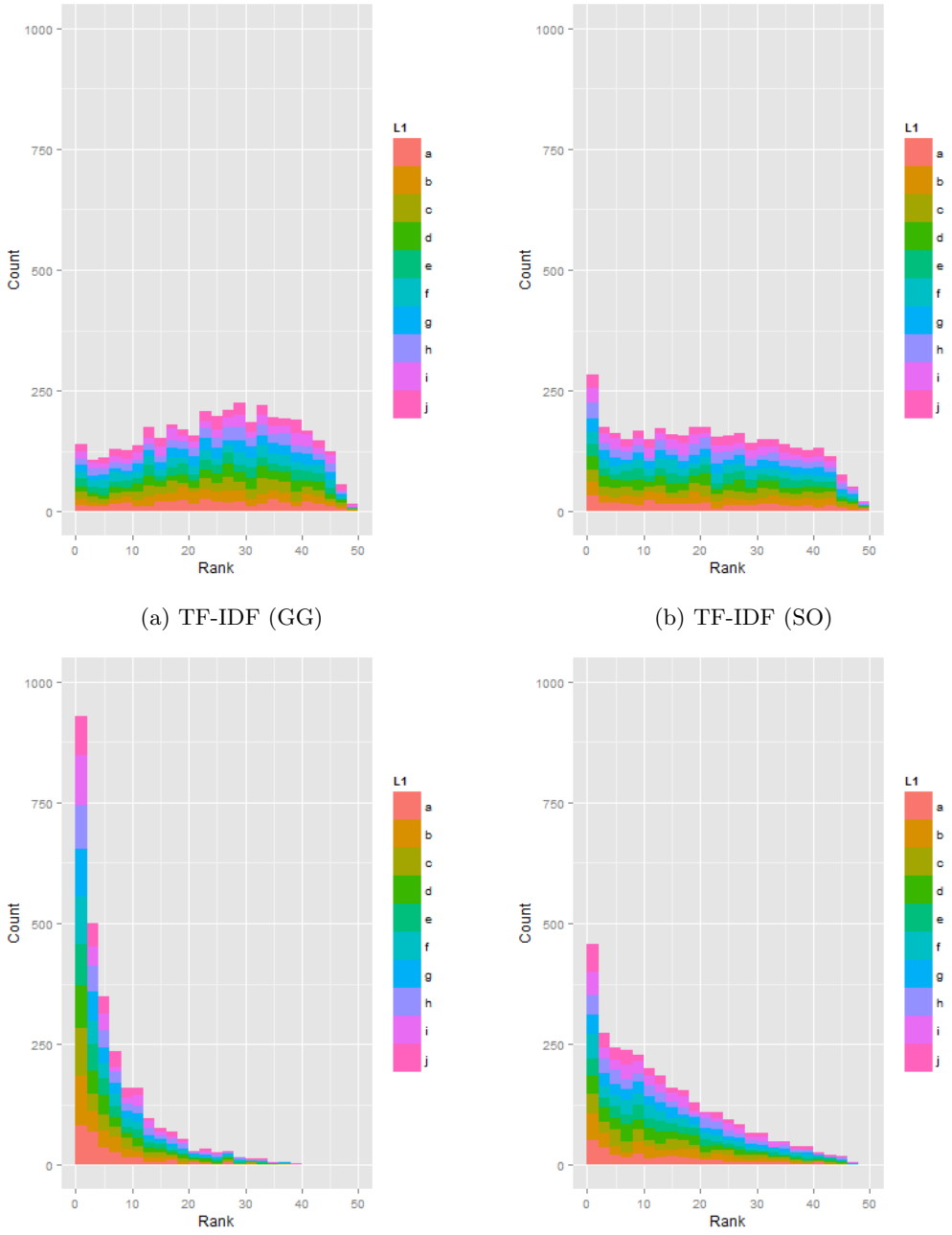
Although the TF-IDF approach is able to analyse the most questions the Custom approach performs better. For both communities the Custom approach has a majority of the rankings below 10, Figure 4.1, and is therefore used to make a selecting of questions for the survey discussed in Section 4.2.2. The visualisation of the results in Figure 4.1 represent on the x-axis the median rank of expert users participating in a question. The y-axis indicates the number of times a specific rank occurs. The different tests of the 10-fold cross validation are distinguishable by their color.

4.2.2 Survey

Although a large part of our approach can be tested using the automated verification it is not able to verify everything. That is, the users that have the required expertise to answer a question but happen not to have answered it. To verify this remaining part a survey is created, the survey is held under members of the ggplot2 community on Google Groups. The purpose of this survey is to verify if the approach used in this project is able to find the correct experts to answer questions on Q&A communities, i.e., Google Groups, Stack Overflow, and GitHub. Although GitHub is not a Q&A community the issues of GitHub projects can be interpreted the same as questions on Q&A communities and users with a matching expertise can be recommended in a similar way. Each survey is personalized for a specific participant and contains 20 questions that are selected using the Custom approach. The questions are equally divided among the categories, which are defined as follows:

1. Google Group questions that are recommended by the approach for a user, and that user also did answer the question.
2. Google Group questions that are **NOT** recommended by the approach for a user, but that user did answer the question.
3. Google Group questions that are recommended by the approach for a user, but that user did **NOT** answer the question.
4. Google Group questions that are **NOT** recommended by the approach for a user, and that user also did **NOT** answer the question.
5. GitHub issues that are recommended by the approach for a user.

The survey participants are asked to answer for each question whether or not they have the required expertise to answer that question. We expect survey participants to agree with all questions from the categories 1 and 2 as those are the questions they participated in. However, that would mean that the algorithm by definition does not work for questions of category 2. Categories 3, 4, and 5 are of more interest as those are the questions they did not participate in. Like the survey on the general expertise this survey was also announced in the ggplot2 community on Google Groups. The response rate for this survey was rather low, in five weeks time we received three responses for which two surveys could be created. Although the third participant is an expert in ggplot2 and occasionally gives workshops about ggplot2 he was not very active on Google Groups, there were only a total of two answers in the dataset. Because the expertise of users is defined by their activity in the community it was not possible to select questions matching his expertise. Even though the response rate was low the survey gave some useful insight. The



(a) TF-IDF (GG)

(b) TF-IDF (SO)

(c) Custom (GG)

(d) Custom (SO)

Figure 4.1: Expertise ranking results.

results of the survey are available in Table 4.4. Note that category 2 for both participants is very small, 0.4% and 0.06% of the total amount of questions. Category 2 contains therefore only 3 instead of 4 questions for one participant, resulting in 7 questions in total.

	Category 1	Category 2	Category 3	Category 4	Category 5
Agree	8	7	7	7	1
Disagree	0	0	1	1	7
Total	8	7	8	8	8

Table 4.4: Survey responses focussed expertise.

The categories can be divided in two groups, the first group contains questions a user participated in and the second group contains only questions a user did not participate in. For group 1, consisting of categories 1 and 2, all participants agreed they had the required expertise to answer the questions. A perfect result for our algorithm would be to have all agree in category 1 and all disagree in category 2. This difference is however as we expected, otherwise the participants would not have been able to answer the questions. However, for category 2 this means the questions were not recognized by the algorithm to be a match with the participant's expertise. A possible explanation for the questions in category 2 is that the word list used to create the document- and expertise vectors is incomplete. Currently the word list is based on the ggplot2 documentation and although we could analyse only 65% during cross validation, when using the complete data set for training to represent a more practical use case 83% can be analysed, results available in Appendix C.4. An incomplete word list is one of the reasons the remaining 17% can not be analysed as mentioned in Section 4.2.1.

The second group contains only questions users did not participate in and consist of categories 3, 4, and 5. A perfect score for the algorithm would be to have all agree in categories 3 and 5, and all disagree in category 4. Our results show for categories 3 and 4 that the participants agreed they had the required expertise for 7 out of 8 questions. A near perfect result for category 3 but for category 4 we expected the participants not to agree. The unexpected results of category 4 are likely caused by the same issue as category 2, an incomplete document- and expertise vector. The unexpected results of categories 2 and 4 will be discussed in more detail later. The final category of this group is used to test if the ggplot2 expertise data found by analysing Google Groups and Stack Overflow can be used for GitHub issues as well. The idea is not to find ggplot2 users who can solve the issue but to find users who are familiar with the feature(s) mentioned in the issue and can assist the developers to track down the problem. For example by giving additional information, discussing the cause of the problem, or giving a minimal reproducible example. The results show that only for 1 out of 8 questions the participants agreed they had the required expertise, indicating that this approach is not suitable for all GitHub issues. Some issues are too technical and intended for developers only, there are however issues that require reproducible examples or tests on different systems, for those type of issues the community can help. One of the participants also mentioned as feedback he would be able to help with issues where reproducible examples or testing is required.

The unexpected results for the categories 2 and 4 required further analysis. After inspecting the questions selected for those categories a flaw was detected in our approach. The way we interpreted the lack of expertise of a user was incorrect. This flaw is best explained using the document vectors $Doc1$ and $Doc2$, and expertise vector $User1$ defined in Equation 4.6.

$$Doc1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, Doc2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, User1 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{4.6}$$

Assume $User1$ did not answer the questions described by $Doc1$ and $Doc2$. When comparing $Doc1$ with the expertise of $User1$ there will be no match because $User1$ does not have the required expertise to answer that question. The question for this user is therefore placed in category 4.

This works fine and is what we intended the algorithm to do. When comparing $Doc2$ and $User1$ there is also no match and the question is therefore placed in category 4 as well. However, this causes a problem because the comparison has nothing to do with the user's expertise. The 'empty' document vector only means that the document did not contain any known terms. A solution for this flaw is discussed in Section 4.3.

4.3 Improvements

The problem with our first approach was that we did not interpreted the lack of expertise correctly. To solve this problem an additional requirement is added for categories 2 and 4. That is, every question should have a document vector with at least one known term.

Another problem with the current approach is the limited word list. A possible expansion is adding the arguments as well for all the functions that are currently in the word list. For a complete word list a thorough examination of all questions in the categories 2 and 4 will be necessarily. This will be interesting to look into as future work.

There is however a small improvement that can be made in the current word list. There are a number of terms that are too general, appear in almost all questions, or do not relate to a specific feature. Those are qplot (16%), ggplot (78%), ggplot2 (51%), and aes(48%). The percentages indicate the amount of questions they appear in.

An inspection of the question count in each of the categories resulted in the following tables, Table 4.5 shows the original counting and Table 4.6 the question count for the improved version, i.e. without the above mentioned general terms and a new interpretation for lack of expertise. Note that the removal of terms influences the question count for all categories, e.g. the question count in category 2 for participant 3 increases from 0 to 1 because that question no longer contains any terms from the document vector.

	Participant 1	Participant 2	Participant 3
Category 1	187	9	2
Category 2	18	3	0
Category 3	4198	4059	3918
Category 4	768	1101	1251
Category 5	828	755	668

Table 4.5: Question count expertise categories (Original)

	Participant 1	Participant 2	Participant 3
Category 1	145	8	1
Category 2	3 (60)	1 (4)	1
Category 3	3216	2469	718
Category 4	164 (1762)	1050 (2753)	2808 (4626)
Category 5	667	484	64

Table 4.6: Question count expertise categories (Improved)

The values between parenthesis are the question count when not ignoring 'empty' document vectors and is added to show what effect the new interpretation of lack of expertise has. The difference between the normal count and the count between parenthesis indicates the amount of discarded questions. The removal of the terms mentioned before is especially noticeable in category 3 because it effects the terms users are familiar with. The recommended questions for participant 3 for example dropped from 3918 to 718, a decrease of more than 80%. This is mostly caused by the removal of 'ggplot' which occurs in 78% of all questions. Ignoring the 'empty' document vectors has effect on questions from categories 2 and 4. Besides the requirement that a user is not familiar with the terms used in those questions, each question should contain at least one term that is mentioned in the word list.

To validate the changes we asked our two participants to complete a final survey. The survey consist of questions from categories 3 and 4, for each category we selected five questions. The reason only questions from categories 3 and 4 are selected is because those are the categories with the most significant changes. For category 5, the GitHub issues, the previous survey already indicated its limitations. Categories 1 and 2 are only slightly changed and do not contain any new questions, in addition category 2 is now so small it does not contain enough questions for a survey.

	Category 3	Category 4
Agree	9	4
Disagree	1	6
Total	10	10

Table 4.7: Survey responses focussed expertise.

Table 4.7 contains the survey results of our improved approach. A perfect result for our algorithm would be to have all agree on category 3 and all disagree on category 4. For category 3 this means our approach performed conform our expectations, i.e. the participants agreed they had the required expertise 9 out of 10 times. For category 4 we already notice improvements, the participants now disagree on 6 out of 10 questions while for the original approach participants disagreed only on 1 out of 8 questions. We believe further improvements are possible by expanding the limited word list used for the document- and expertise vector as it will allow us to describe questions and user's expertise in more detail.

4.4 Conclusion

The focussed expertise is implemented to expand the general expertise approach. By combining the approaches we can tell for expert users on what specific aspects of ggplot2 they are knowledgeable. This allows us to find matching questions and issues for a user's expertise. Furthermore we wanted to achieve two things. First of all we wanted to compare different text mining and analysing techniques to be able verify what technique(s) are suitable for describing a user's expertise. Secondly, to verify if the resulting expertise data can be used to recommend questions and issues that match a user's expertise.

From the comparison between TF-IDF and our custom approach we can conclude the custom approach is best suited to describe questions and users' expertise. The most important difference being that with TF-IDF you have no control over the words used in the document- and expertise vectors while by using the custom approach you are free to choose what words are used. By using the ggplot2 documentation as base for the word list the approach accurately describes questions and users' expertise. To be able to find good matches between questions and users expertise we compared different representations of the expertise vectors, e.g. expertise vectors with/without term frequency and irrelevant terms. By matching the document- and expertise vectors of different representations, as shown in Equation 4.4 and 4.5, we can conclude that vectors without term frequencies and irrelevant terms are best suited to find accurate matches.

Although the first results of the question recommendations did not met our expectations the improved version came closer. The participants could answer 9 out of 10 questions recommended by the algorithm. However, questions interpreted by the algorithm as no match for the expertise of a user were only 6 out of 10 times correct. By expanding the word list we believe this could be improved further.

Besides the question recommendations we also looked into GitHub issues. Based on the survey results and participants' feedback the GitHub issue recommendations are in its current state of limited use. Even thought a user is familiar with a specific term used in a GitHub issue the description is often too technical and intended for developers instead of users. There are however issues that require reproducible examples, additional feedback or tests on different systems, for those type of issues the community can help.

Chapter 5

Related Work

There are multiple research areas relevant to this work, (5.1) data mining, (5.2) identity merging, (5.3) expertise identification, and (5.4) topic analysis. The different areas of related work will be discussed in order of their occurrence in our work.

5.1 Data Mining

Analysing online communities requires a lot of data. Collecting, sorting, and analysing data has however its challenges. In a paper by Kalliamvakou et al. [17] such perils are discussed for GitHub. A number of interesting perils mentioned are: (I) Most projects are inactive, (II) A large portion of repositories is not for software development, (III) Many active projects do not conduct all their software development on GitHub. The project selected for our use-case, ggplot2, does not suffer from those perils. Ggplot2 is an active (I) software project (II) that uses GitHub as their main development platform (III) and is therefore a suitable project for our study.

Some communities offer data dumps themselves (Stack Overflow), for other communities third parties have created them (GitHub), and in other cases the researchers themselves will have to create them (Google Groups). GitHub offers some data through their API but for a complete data dump you are dependant on a third party. In our study we use the GHTorrent project, introduced by Gousios et al. [12], as an effort to create a scalable, queriable, offline mirror of data offered through the Github REST API. Making it easier for the research community to access and analyse GitHub data. A related work to this is Lean GHTorrent [13], an addition to GHTorrent which makes it possible to request customizable data dumps on demand through a web interface.

5.2 Identity Merging

In our study multiple communities are combined to find the expertise of users. In order to successfully combine them users active in multiple communities need to be identified and merged. There are a number of techniques available to realise this. The preferred technique depends on the available information and can differ per community. In our work we used an approach by Vasilescu et al. [37], which is especially useful when merging identities across multiple communities. The approach works by assuming an email address belongs to a single user. The advantage of using email addresses in contrast to other techniques is that you do not have any false positives. Vasilescu et al. uses this approach to merge identities from Stack Overflow and GitHub. Although Stack Overflow does not make email addresses public a MD5 hash is available. By creating a MD5 hash for GitHub email addresses as well it is possible to merge identical users. Note that since 2014 Stack Overflow no longer makes the MD5 hashes public. Therefore it is required to merge an older dataset (March 2013), that contains MD5 hashes, with the current dataset to be able to use the latest data.

Although in our approach identities are merged by only using email addresses a common used alternative is merging by name [7][11][18]. Merging identities does not always happen across multiple communities, Teyton et al. [35] mentions for some CVS like systems users are free to choose their name at every new revision they commit. Over time this will result in multiple variations of a name that all belong to the same identity. This is however not relevant for GitHub as each user has a unique id. The reason we choose to merge identities using email addresses instead of names is that names are not unique. Common names especially will often result in false positives when merging identities, email addresses on the other hand are unique and will not have this issue.

Goeminne et al. [11] compare multiple identity merge algorithms that use user names. In addition they introduce an algorithm based on the work by Bird et al. [7] and Robles et al. [31] that works as follow. First names are normalized, i.e. removing accents, converting to lower case, replacing multiple whitespace characters by a single space, and removing leading and trailing whitespaces. The similarity of identities is determined by the so called thresholded Levenshtein distance used by Bird et al. From the approach by the Robles et al. the email prefix comparison is used. That is, a list of possible prefixes is created based on a users name. When a match is found within the list the identities are merged. The prefixes are of the form firstname.lastname, lastname.firstname, or a combination of the first name followed by the first letter of the last name. Another paper on identity merging, by Kouters et al. [18], uses Latent Semantic Analysis (LSA) to merge similar aliases (name-email tuples). Similar to our approach they also use email addresses but combine this with names and also support variations of email addresses. The advantage of using LSA is that it is robust against more types of differences in aliases and reduces the amount of noisy data, i.e. aliases used by the same individual with illogical variations.

Although merging by name has disadvantages it is interesting as future work if combined with the available email addresses. It could merge identical users that use different email addresses for each community or merge identical users for which we do not have an email address at all.

5.3 Expertise Identification

There are many approaches available to measure a user's level of expertise in communities, e.g., approaches based on social circle [42], activity [35][38], or behaviour [26][27][28]. An extended list of related work on expert identification can be found in Appendix B.1. Our approaches are activity based, one approach measures the overall expertise by using the question and answer count, the other analyses questions and answers to measure whether or not a user has expertise for a specific topic. The reason our approach is activity based is because it feels logical to define users' expertise based on their activities and contributions they make within a community. Their activities and contributions show for a fact what a user knowns and is capable of. Another work focussed on activity is that of Venkataramani et al.[38]: instead of analysing text from questions and answers their approach analyses source code to measure the topics of expertise of users. The technical terms found in the source code are then mapped to tags used for Stack Overflow questions. This allows them to recommend users to answer specific questions. The approach by Teyton et al. [35] analyses source code as well but focusses only on used libraries to be able to recommend experts per library.

Besides using the expertise results to find experts for answering questions on Q&A communities it has also potential for finding help for unsolved issues on code hosting communities like GitHub or for bug triaging in general [4][6]. Anvik et al. [4] identifies experts to semi-automate a part of the bug triaging process, the assignment of developers to bug reports. Using machine learning techniques recommendations for a bug report are made based on bug reports that the developers have previously solved. This results in a list of recommended developers that the triager can use to select and assign a developer to a bug report. A more recent work focussing on bug triaging is that of Badashian et al. [6]. Similar to our approach Badashian et al. uses expertise data across communities, Stack Overflow and GitHub, to recommend users for bug reports. Questions with specific tags are analysed on Stack Overflow and GitHub is used to analyse if a user has solved

any bugs recently, this information is then used to define a user's expertise and to recommended users for unsolved bug reports. A difference with our approach is that a user's expertise is defined using multiple sources, furthermore our work focussed more on unanswered questions in Q&A communities instead of recommendations bug triaging.

Approaches not based on activity might be less specific when defining the expertise of users and are more comparable to our general expertise but they are nonetheless interesting approaches. Pal et al. [26][27][28] uses behaviour of users to identify experts. One work focusses on the motivation and ability of users to help others [26], the other studies use selection bias to identify experts [27][28]. The idea behind selection bias is that experts prefer to answer specific questions, i.e. questions where they have a higher chance of making a valuable contribution. Their proposed mathematical model to estimate this selection bias allows them to identify experts.

Another relevant activity is to verify if the expert findings are indeed correct. Previous works have verified expertise algorithms using different approaches such as manual inspections [29], or human raters [26][42]. For our study we have combined human raters with surveys, making it easy accessible for the members of the ggplot2 community to participate in the verification process. In addition we automated a part of our verification process which is discussed in detail in Section 4.2. A previous work using human raters is from Pal et al. [26], their expertise findings in the TurboTax Live Community (TTLC) are verified by hand by employees of Intuit, the company behind TTLC. Zhang et al. [42] uses a similar approach by using two known Java programming experts to rate a selection of Java users from 1 (Newbie) to 5 (Top Java expert). The results are then compared against the algorithm rankings.

A study by Pratkanis [29] uses a manual inspection to verify results. His work analyses the Robot Operating System (ROS) community, to identify experts and sub-communities within ROS. The top users who were identified using PageRank are verified by checking if they are listed as top contributor on ROS. This approach can however not be applied on our work because only Stack Overflow has some kind of expert recognition based on reputation. The other communities, Google Groups and GitHub, do not have such information available.

5.4 Topic Analysis

Text mining or topic analysis can give useful information about the content of documents, it can for example help categorize documents, analyse the topics of interest of authors, or find popular topics. In our study we use it to analyse the questions and answers posted by users to be able to find the topics users are familiar with. Besides being applied on textual documents [14][41], text mining is also used for log files [3] and source code [38]. One of the methods used in our work is TF-IDF, which is a common approach to represent documents (questions and answers) using vectors. TF-IDF and its more basic variant, sparse Bag of Words (sBoW), is discussed by Cheng et al. [9]. Cheng et al. also introduces a new method the so called Dense Cohort of Terms (dCoT) as alternative for TF-IDF and sBoW. A disadvantage of sBoW is the sparsity, dCoT addresses this by mapping a high-dimensional sparse vector into a more dense vector by replacing infrequent words by more frequent used words. For example the rare words *splendid*, *spectacular*, *terrific*, and *glorious* are all replaced by the frequent word *good*. Although this approach is good for textual documents and reduces the dimension of vectors it is not suited for our approach. That is because our vectors are based on functions found in the documentation of ggplot2, combining functions will only result in a less specific description of a user's expertise.

Chapter 6

Tool Chain

During this project multiple tools are developed for collecting, parsing, and analysing data. The purpose of this tool chain is to make future expertise analysis easier by introducing a step-by-step approach. This section describes the purpose of each tool and how they are related to each other. For a complete overview of the tool chain see Figure 6.1. The source code is available on GitHub[1].

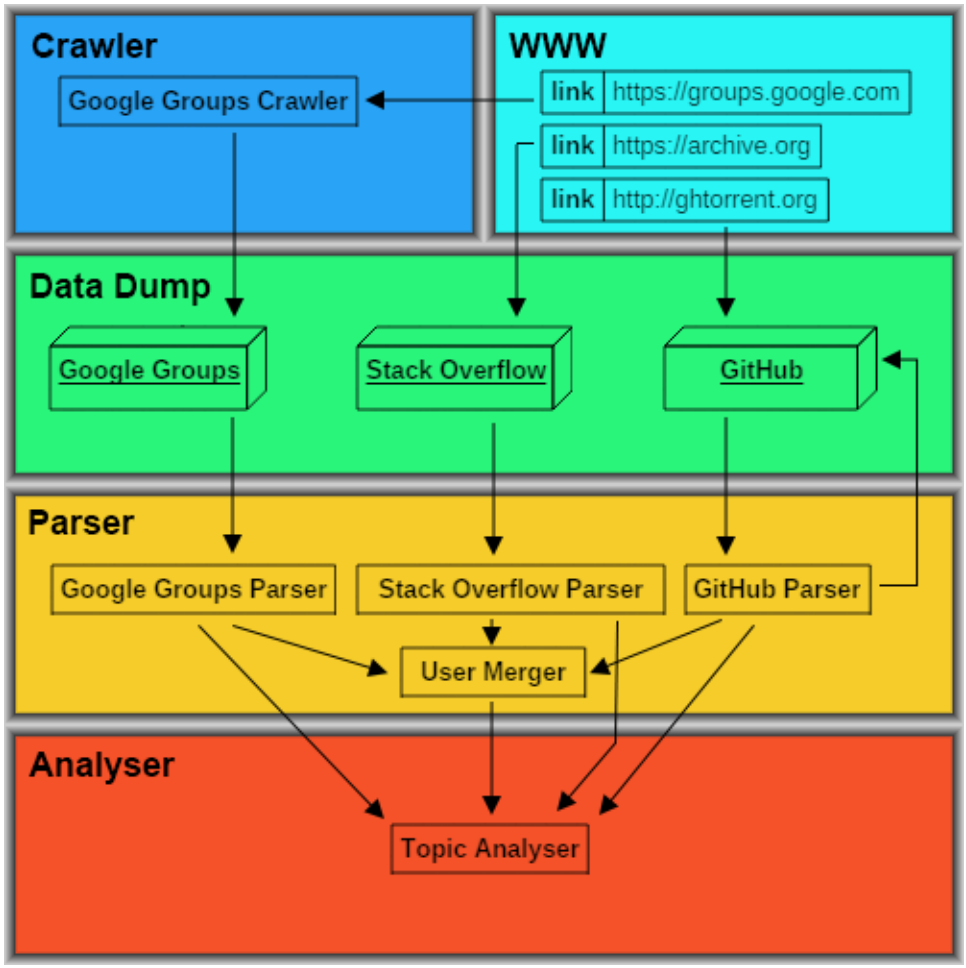


Figure 6.1: Tool chain overview.

[1] https://github.com/josdemmers

6.1 Data Crawler

The first step in the tool chain is to collect all the required data. For both Stack Overflow and GitHub a data dump is available for download, respectively at the Internet Archive and GHTorrent, therefore only a data crawler is developed for Google Groups.

Google Groups Crawler	
Input	GG_useremaillist.txt
Output	GG_topiclist.txt <topic id>.xml
Description	Data crawler for Google Groups. Creates a xml file for every single topic posted on Google Groups for a given group.

Table 6.1: Tool summary for Google Groups Crawler.

The Google Groups Crawler, Figure 6.2, collects all topics of a given Google Groups community, ggplot2 in this case. Each topic is saved in a separate xml file and contains all related posts, i.e., the question and all given answers. Furthermore the list of topics is saved in GG_topiclist.txt.

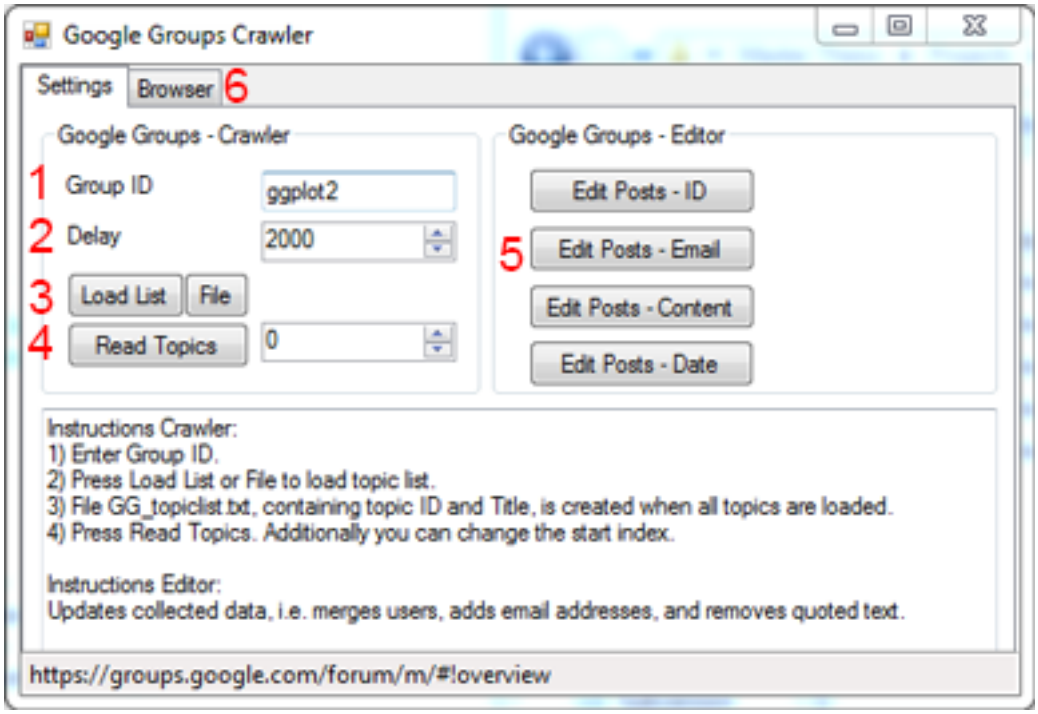


Figure 6.2: Interface Google Groups Crawler.

1 Name of the Google Group the data is collected from.
2 Delay in ms between http requests.
3 Load a list of all topics, either by creating a new list from the Google Group (Load List) or by loading an existing file (File), GG_topiclist.txt.
4 Export topics listed in GG_topiclist.txt starting from the given index. Each topic is saved as a separate XML file.
5 Modifies exported topics, i.e., merges users with multiple identities, adds email addresses, removes quoted content, and fixes the date format.
6 The browser is used to collect all data and it shows the crawlers progress.

6.2 Data Parser

The second step in the tool chain is to parse the collected data. Tools were created for Google Groups, Stack Overflow, and GitHub.

Google Groups Parser	
Input	GG_topiclist.txt <topic id>.xml
Output	GG_users.xml GG_topics.xml GG_stats.xml GG_stats.csv GG_useremaillist.txt
Description	Data parser for Google Groups. Combines the individual topic files into single xml files for users, topics, and statistical data.

Table 6.2: Tool summary for Google Groups Parser.

The Google Groups Parser, Figure 6.3, processes the data collected with the Google Groups Crawler. The individual XML files created by the crawler are converted into four categorized XML files. Those categories are users, topics, user statistics, and e-mail addresses.

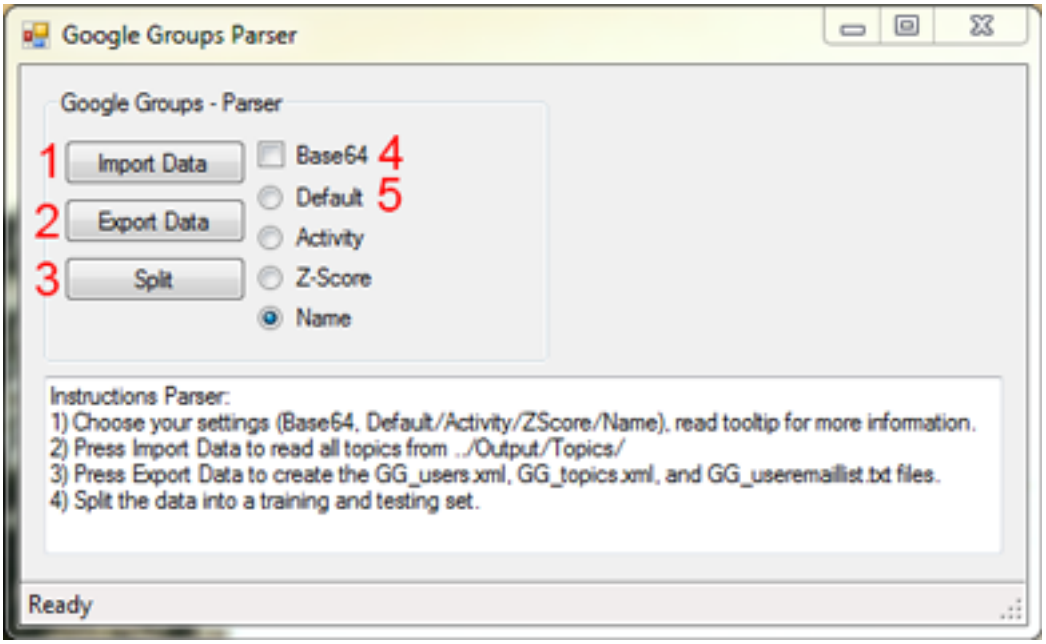


Figure 6.3: Interface Google Groups Parser.

1 Import the data collected by the Google Groups Crawler.
2 Export the data to four categorized xml files, users, topics, user statistics and e-mail addresses.
3 Split the data set in 10 training and testing subsets.
4 Option to encode posts using Base64.
5 Option to sort users by Activity, ZScore, or Name. Note: the order of users influences which user Id is assigned to each user. To replicate this study sort users by name.

The Stack Overflow Parser, Figure 6.4, processes the data dump and filters posts and users by a given tag, ggplot2 in this case. The user file is created in two steps. First of all the users are

Stack Overflow Parser	
Input	Posts.xml Users_2013-3.xml Users.xml
Output	SO_Posts_filter.xml SO_Users_filter.xml SO_Users_filter_hash.xml SO_stats.xml SO_stats.csv
Description	Data parser for Stack Overflow. Filters the Stack Overflow data dump and saves them in multiple smaller xml files for further analysis.

Table 6.3: Tool summary for Stack Overflow Parser.

filtered such that only the users that asked or answered questions for a given tag remain. Secondly the user file is updated with email hashes from the March 2013 data set because the current data set no longer contains email hashes. The email hashes are important further on the tool chain for identity matching across multiple communities. In addition to the filtered post and user files a file containing statistical information is created.

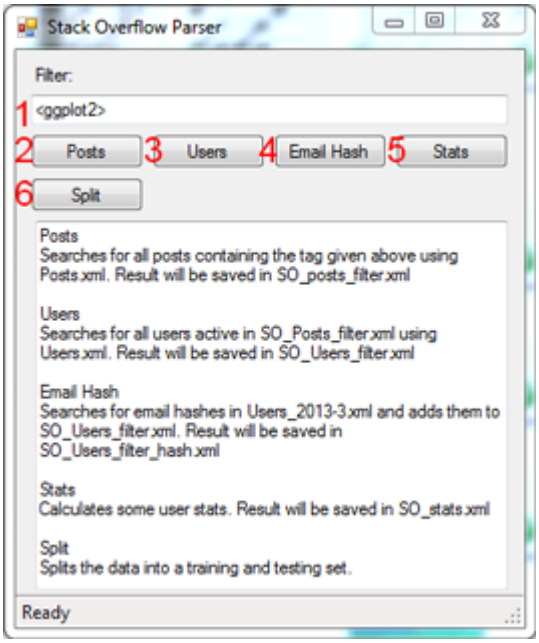


Figure 6.4: Interface Stack Overflow Parser.

1 Stack Overflow tag used to filter the posts and users.
2 Exports all posts in SO_Posts_filter.xml with the given tag.
3 Exports all users in SO_Users_filter.xml who either have asked or answered a question with the given tag.
4 Updates SO_Users_filter.xml with email hashes, result is saved in SO_Users_filter_hash.xml. Note: this is a time consuming task and will take around five hours to complete.
5 Exports statistical data of users for future analysis as SO_stats.xml and SO_stats.csv.
6 Split the data set in 10 training and testing subsets.

The GitHub Parser, Figure 6.5, processes the data dump from GHTorrent[12]. The data dump contains user, issue, and issue-comment data for ggplot2. As the GHTorrent project offers the

GitHub Parser	
Input	github_ggplot2_issues_json.txt github_ggplot2_issue-comments_json.txt github_ggplot2_users_json.txt
Output	GH_users_json_query.txt GH_users_json_fixedformat.txt GH_issues_json_fixedformat.txt GH_issue-comments_json_fixedformat.txt GH_users.xml GH_posts.xml GH_stats.xml GH_stats.csv
Description	Data parser for GitHub. Converts data dump from GHTorrent from JSON to XML and creates multiple XML files for further analysis.

Table 6.4: Tool summary for GitHub Parser.

data through a MongoDB server a MongoDB client is required to access the data, e.g., Mongo Management Studio[2]. The query to retrieve the issue and issue-comment data is available in Listing 6.1 and Listing 6.2.

```
db.issues.find({repo:"ggplot2", owner:"hadley"})
```

Listing 6.1: JSON query for retrieving ggplot2 issues data.

```
db.issue_comments.find({repo:"ggplot2", owner:"hadley"})
```

Listing 6.2: JSON query for retrieving ggplot2 issue-comment.

The data offered is in the JSON format and is converted to XML by the parser. Resulting in XML files for posts, users, and user statistics. Note that posts in this context are issues and issue comments.

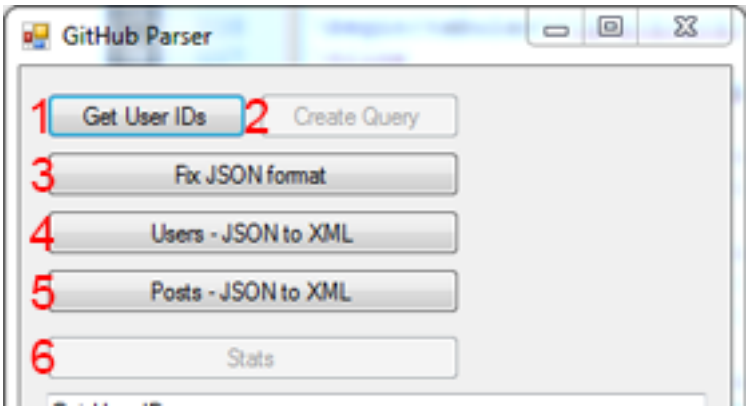


Figure 6.5: Interface GitHub Parser.

1 Searches for all users who created or commented on an issue using github_ggplot2_issues_json.txt and github_ggplot2_issue-comments_json.txt as input

2 Creates a JSON query for MongoDB with the users found in step 1. Button is enabled when step 1 has been completed. The query is used to get a data dump of all users.

[2]http://www.litixsoft.de/english/mms/

3 Fixes the JSON format of github_ggplot2_issues_json.txt, github_ggplot2_issue-comments_json.txt, and github_ggplot2_users_json.txt.
4 Converts GH_users_json_fixedformat.txt from JSON to XML, result is saved as GH_users.xml.
5 Converts GH_issues_json_fixedformat.txt and GH_issue-comments_json_fixedformat.txt from JSON to XML, result is saved as GH_posts.xml.
6 Exports statistical data of users for future analysis as GH_stats.xml and GH_stats.csv. Button is enabled when the user and post data has been parsed.

User Merger	
Input	SO_Users_filter_hash.xml GG_users.xml GH_users.xml *_stats.xml
Output	Users_merged_SO.xml Users_merged_SO_GG.xml Users_merged_SO_GG_GH.xml
Description	Merges user data of the Stack Overflow, Google Groups, and GitHub communities.

Table 6.5: Tool summary for User Merger.

The User Merger, Figure 6.6, merges user data of the Stack Overflow, Google Groups, and GitHub communities.

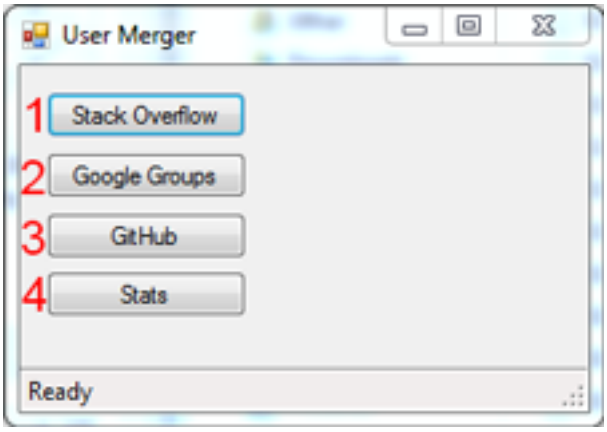


Figure 6.6: Interface User Merger.

1 Prepares Stack Overflow data for user merge, result is saved in Users_merged_SO.xml.
2 Merges the Stack Overflow and GitHub users, result is saved in Users_merged_SO_GG.xml.
3 Adds users from GitHub to the merge, result is saved in Users_merged_SO_GG_GH.xml.
4 Adds user's ZDegree to the merge, for users active in multiple communities the highest ZDegree is added. Result is saved in Users_merged_SO_GG_GH_stats.xml. The ZDegree data is used by the Topic Analyser tool.

6.3 Data Analyser

The third and final step in the tool chain is analysing the data. Besides the developed tools there are also a number of R scripts created for the data analysis.

General Expertise

The R script, summarized in Table 6.6, is used to measure the correlation of ranking between the different general expertise algorithms discussed in Chapter 3. The input files are manually created using the GG_stats.csv and SO_stats.csv files created by the data parsers. The script does not create any new files its only purpose is to calculate the correlation. Furthermore R scripts are used in the general expertise analysis by visualizing the data in graphs, Table 6.7 and 6.8.

R: ggplot2_alg_correlation.R	
Input	GG_stats_correlation.csv GG_stats_correlation - Top50.csv SO_stats_correlation.csv SO_stats_correlation - Top50.csv
Output	
Description	R script for measuring the correlation of rankings between the different expertise algorithms.

Table 6.6: Script summary for ggplot2_alg_correlation.R.

R: ggplot2_barplot.R	
Input	GG_stats.csv
Output	
Description	R script for plotting a bar plot for post count.

Table 6.7: Script summary for ggplot2_barplot.R.

R: ggplot2_lorenz_curve.R	
Input	GG_stats.csv
Output	
Description	R script for plotting a lorenz curve of the inequality of the amount of answers given by users.

Table 6.8: Script summary for ggplot2_lorenz_curve.R.

Focussed Expertise

The remaining R scripts are for the focussed expertise analysis discussed in Chapter 4. The output of the scripts in Table 6.10, 6.11, and 6.12 are used by the Topic Analyser tool, Figure 6.7, to create expertise vectors. Furthermore the script in Table 6.13 is used to measure the quality of the focussed expertise algorithms.

The Topic Analyser is used to measure and verify the focussed expertise of users for both Google Groups and Stack Overflow. The focussed expertise of users is measured using two different methods. The first one uses (GG,SO)_wordlist_tfidf.csv as input and creates expertise vectors based on terms found in user's posts using TF-IDF. The second one uses ggplot2_wordlist.csv as input and creates expertise vectors based on a custom word list created from functions found in the ggplot2 documentation. For users active in Stack Overflow and Google Groups both communities are used to create an expertise vector. The Topic Analyser is set to measure the focussed expertise of the top 50 ranked users by the ZDegree algorithm, for which we used Users_merged_SO_GG_GH_Stats.xml to filter the users.

The user and post/topic XML files are used to know which users were active in a topic, i.e., posted an answer. This information is used by the tool when verifying the focussed expertise of users. The results of the verification are saved in (GG,SO)_Custom_avg_rank.csv and (GG,SO)_TFIDF_avg_rank.csv. In addition to the automated verification process there is also a manual verification as discussed in Chapter 4, for which the Topic Analyser is used to create a list of possible questions that can be used in the survey.

Topic Analyser	
Input	GG_users.xml SO_Users_filter_hash.xml GG_topics.xml SO_Posts_filter.xml Users_merged_SO_GG_GH_Stats.xml (GG,SO)_userlist.csv (GG,SO)_wordlist_tfidf.csv (GG,SO)_questionlist.csv ggplot2_wordlist.csv
Output	(GG,SO)_Custom_avg_rank.csv (GG,SO)_TFIDF_avg_rank.csv
Description	Topic Analyser for Google Groups and Stack Overflow to measure the focussed expertise of users.

Table 6.9: Tool summary for Topic Analyser.

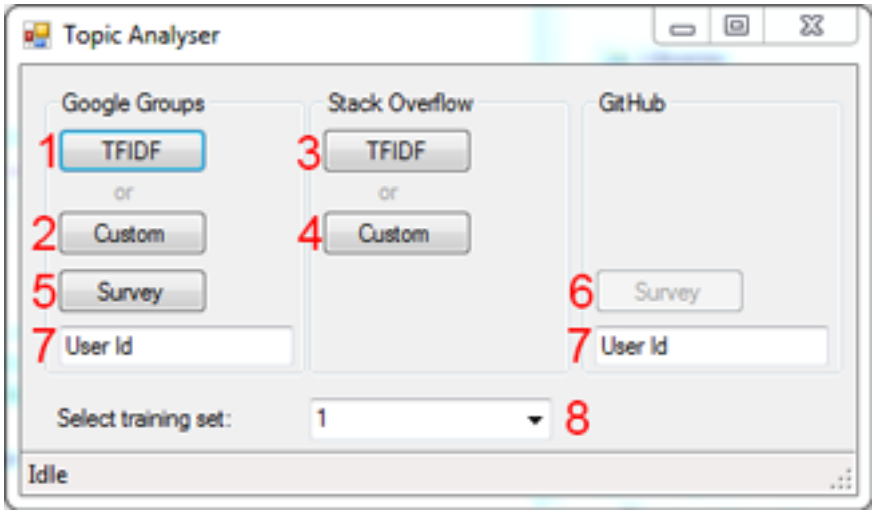


Figure 6.7: Interface Topic Analyser.

1 Create expertise vectors using the TF-IDF terms. Furthermore the expertise vectors are used to find questions matching the users expertise. Finally the quality of the matching is measured.
2 Create expertise vectors using a custom wordlist based on the function names of ggplot2. Furthermore the expertise vectors are used to find questions matching the users expertise. Finally the quality of the matching is measured.
3 Create expertise vectors using the TF-IDF terms. Furthermore the expertise vectors are used to find questions matching the users expertise. Finally the quality of the matching is measured.
4 Create expertise vectors using a custom wordlist based on the function names of ggplot2. Furthermore the expertise vectors are used to find questions matching the users expertise. Finally the quality of the matching is measured.
5 Creates a list of possible questions that can be used in the survey for the user specified in (7).
6 Creates a list of possible questions that can be used in the survey for the user specified in (7). The button is enabled when the Google Group questions for the survey are listed.
7 Both User Id fields are identical and specify the user for who you want to create a list of survey questions.
8 Select a starting point for the testing sets (1-10). The analyser will continue automatically until all 10 sets are analysed.

R: ggplot2_tfidf.R	
Input	GG_topics.xml SO_Posts_filter.xml
Output	GG_wordlist_tfidf.csv SO_wordlist_tfidf.csv
Description	R script for calculating the TF-IDF terms.

Table 6.10: Script summary for ggplot2_tfidf.R.

R: ggplot2_wordlist_functions.R	
Input	http://docs.ggplot2.org/current/
Output	ggplot2_wordlist.csv
Description	R script for exporting all function names from the ggplot2 documentation.

Table 6.11: Script summary for ggplot2_wordlist_functions.R

R: ggplot2_questions.R	
Input	GG_topics.xml SO_Posts_filter.xml GH_posts.xml
Output	(GG, SO, GH)_userlist.csv (GG, SO, GH)_questionlist.csv (GG, SO, GH)_topicIDlist.csv
Description	R script for cleaning / filtering question text. Used by the Topic Analyser to create document vectors.

Table 6.12: Script summary for ggplot2_questions.R.

R: ggplot2_topicexpertise_quality.R R: ggplot2_topicexpertise_quality_visualisation.R	
Input	GG_TFIDF_avg_rank GG_Custom_avg_rank
Output	
Description	R script for measuring the quality of the focussed expertise algorithms.

Table 6.13: Script summary for ggplot2_topicexpertise_quality(_visualisation).R.

Chapter 7

Conclusions

In this thesis, we described a method that uses online communities such as Stack Overflow and Google Groups to create an expertise profile for ggplot2 users active in those communities. This information is then used to recommend users for questions or issues on either Stack Overflow, Google Groups, or GitHub.

By implementing the general expertise (Chapter 3) we defined an overall level of expertise for users to recognize experts in a community. After comparing different expertise ranking algorithms we concluded that ZDegree was best suited for our implementation. In addition the findings from the survey concluded that the algorithm is able to make a clear distinction between regular and expert users. Regarding the focussed expertise (Chapter 4) we concluded that our approach, using a custom word list, performs better than TF-IDF when defining the expertise of users. Furthermore the question recommendation performs reasonable well, especially for questions that match a users expertise. However for questions not matching a users' expertise the algorithm is only correct 6 out of 10 times. For the implementation of the general and focussed expertise we introduced a tool chain which can also be used in future work to parse and analyse Stack Overflow, Google Groups, or GitHub data. In addition we introduce a data crawler for creating Google Groups data sets. Because the Google Groups website changes often the crawler is developed such that it cost minimal effort to update when it is outdated.

7.1 Threats to Validity

An internal validity is threatened by the low amount of participants in our survey, 7 for the survey on general expertise and only 2 for the survey on focussed expertise. Although we were able to make conclusions based on the responses the limited amount of participants has to be taken into account. Regarding construct validity threats, any missing words in the used word list for the expertise vectors and the decision to use word lists as such could give an incomplete view of a users expertise. Similar vectors are also used to describe the required expertise for a question, when a question does not contain any words from the word list a question is discarded, limiting the amount of questions that can be recommended to users. An external validity threat is that some expert users are not active in the online ggplot2 communities making it impossible to create a correct expertise vector. An example of such users are those who use ggplot2 frequently, teach others by giving workshops but seldom answer questions on Stack Overflow or Google Groups. Another concern is how to treat questions that are not real problems that require an expert but are more opinion based or conversations [1]. Although this is not relevant for Stack Overflow because of its strict moderating policy, it is relevant for Google Groups.

7.2 Future Work

As mentioned in Section 4.3 an advantageous future work would be to expand the word list to be able to better describe questions and users' expertise. As shown in Table 4.6 there are questions being discarded because they cannot be described using the current vector. By improving the word list the amount of discarded questions can be reduced resulting in better recommendations. Another future work of interest is to improve the merging of user identities across communities. Currently this is done by matching email addresses, a logical addition would be to check for names as well. Although merging by name has disadvantages, i.e. it can introduce false positives, it makes it possible to merge identical users that use a different email address for Stack Overflow and Google Groups. In addition it allows us to merge identical users for which we do not have an email address at all.

Bibliography

[1] Lada A Adamic, Jun Zhang, Eytan Bakshy, and Mark S Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *Proceedings of the 17th international conference on World Wide Web*, pages 665–674. ACM, 2008. 47, 55

[2] Mobyen Uddin Ahmend, Shahina Begum, Peter Funk, Ning Xiong, and Bo Von Schéele. Case-based reasoning for diagnosis of stress using enhanced cosine and fuzzy similarity. *Transactions on Case-based Reasoning for Multimedia Data*, 1(1):3–19, 2008. 23

[3] Omar Alonso, Premkumar T Devanbu, and Michael Gertz. Expertise identification and visualization from cvs. In *Proceedings of the 2008 international working conference on Mining software repositories*, pages 125–128. ACM, 2008. 35, 54

[4] John Anvik, Lyndon Hiew, and Gail C Murphy. Who should fix this bug? In *Proceedings of the 28th international conference on Software engineering*, pages 361–370. ACM, 2006. 34, 55

[5] John Anvik and Gail C Murphy. Determining implementation expertise from bug reports. In *Mining Software Repositories, 2007. ICSE Workshops MSR'07. Fourth International Workshop on*, pages 2–2. IEEE, 2007. 55

[6] Ali Sajedi Badashian, Abram Hindle, and Eleni Stroulia. Crowdsourced bug triaging. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*. IEEE, 2015. 2, 34, 55

[7] Christian Bird, Alex Gourley, Prem Devanbu, Michael Gertz, and Anand Swaminathan. Mining email social networks. In *Proceedings of the 2006 international workshop on Mining software repositories*, pages 137–143. ACM, 2006. 34

[8] Mohamed Bouguessa, Benoît Dumoulin, and Shengrui Wang. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 866–874. ACM, 2008. 55

[9] Minmin Chen, Kilian Q Weinberger, Fei Sha, et al. An alternative text representation to tf-idf and bag-of-words. *arXiv preprint arXiv:1301.6770*, 2013. 35

[10] Harris Drucker, Donghui Wu, and Vladimir N Vapnik. Support vector machines for spam categorization. *Neural Networks, IEEE Transactions on*, 10(5):1048–1054, 1999. 21

[11] Mathieu Goeminne and Tom Mens. A comparison of identity merge algorithms for software repositories. *Science of Computer Programming*, 78(8):971–986, 2013. 34

[12] Georgios Gousios and Diomidis Spinellis. Ghtorrent: Github's data from a firehose. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 12–21. IEEE, 2012. 1, 7, 33, 40

[13] Georgios Gousios, Bogdan Vasilescu, Alexander Serebrenik, and Andy Zaidman. Lean ghtorrent: Github data on demand. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 384–387. ACM, 2014. 33

[14] Jinwen Guo, Shengliang Xu, Shenghua Bao, and Yong Yu. Tapping on the potential of q&a community by recommending answer providers. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 921–930. ACM, 2008. 35, 54

[15] Benjamin V Hanrahan, Gregorio Convertino, and Les Nelson. Modeling problem difficulty and expertise in stackoverflow. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion*, pages 91–94. ACM, 2012. 2

[16] Huzefa Kagdi, Maen Hammad, Jonathan Maletic, et al. Who can help me with this source code change? In *Software Maintenance, 2008. ICSM 2008. IEEE International Conference on*, pages 157–166. IEEE, 2008. 54

[17] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining github. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 92–101. ACM, 2014. 33

[18] Erik Kouters, Bogdan Vasilescu, Alexander Serebrenik, and Mark GJ van den Brand. Who's who in gnome: Using lsa to merge software repository identities. In *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pages 592–595. IEEE, 2012. 34

[19] Glenn E Littlepage and Amy L Mueller. Recognition and utilization of expertise in problem-solving groups: Expert characteristics and behavior. *Group Dynamics: Theory, Research, and Practice*, 1(4):324, 1997. 2

[20] Max O Lorenz. Methods of measuring the concentration of wealth. *Publications of the American statistical association*, 9(70):209–219, 1905. 12

[21] Bjoern-Elmar Macek, Martin Atzmueller, and Gerd Stumme. Profile mining in cvs-logs and face-to-face contacts for recommending software developers. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 250–257. IEEE, 2011. 54

[22] 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 International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, pages 2857–2866, 2011. 1

[23] Nora McDonald and Sean Goggins. Performance and participation in open source software on github. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 139–144. ACM, 2013. 1

[24] David Meyer, Kurt Hornik, and Ingo Feinerer. Text mining infrastructure in r. *Journal of Statistical Software*, 25(5):1–54, 2008. 22

[25] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999. 9

[26] Aditya Pal, Rosta Farzan, Joseph A Konstan, and Robert E Kraut. Early detection of potential experts in question answering communities. In *User Modeling, Adaption and Personalization*, pages 231–242. Springer, 2011. 2, 34, 35, 54

[27] Aditya Pal, F. Maxwell Harper, and Joseph A. Konstan. Exploring question selection bias to identify experts and potential experts in community question answering. *ACM Trans. Inf. Syst.*, 30(2):10, 2012. 2, 34, 35

[28] Aditya Pal and Joseph A Konstan. Expert identification in community question answering: exploring question selection bias. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1505–1508. ACM, 2010. 2, 34, 35, 54

[29] Tony Pratkanis. Identification of distinct sub-communities, high-value members and significant software packages within the ros (robot operating system) community. 2014. 2, 35, 54

[30] Juan Ramos. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003. 21

[31] Gregorio Robles and Jesus M Gonzalez-Barahona. Developer identification methods for integrated data from various sources. *ACM SIGSOFT Software Engineering Notes*, 30(4):1–5, 2005. 34

[32] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988. 21

[33] Patrick E Shrout and Joseph L Fleiss. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420, 1979. 17

[34] Ahmed Tamrawi, Tung Thanh Nguyen, Jafar Al-Kofahi, and Tien N Nguyen. Fuzzy set-based automatic bug triaging: Nier track. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 884–887. IEEE, 2011. 55

[35] Cédric Teyton, J-R Falleri, Floréal Morandat, and Xavier Blanc. Find your library experts. In *Reverse Engineering (WCRE), 2013 20th Working Conference on*, pages 202–211. IEEE, 2013. 2, 34, 54

[36] Patanamon Thongtanunam, Chakkrit Tantithamthavorn, Raula Gaikovina Kula, Norihiro Yoshida, Hajimu Iida, and Ken-ichi Matsumoto. Who should review my code? a file location-based code-reviewer recommendation approach for modern code review. In *Software Analysis, Evolution and Reengineering (SANER), 2015 IEEE 22nd International Conference on*, pages 141–150. IEEE, 2015. 55

[37] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. Stackoverflow and github: associations between software development and crowdsourced knowledge. In *Social Computing (SocialCom), 2013 International Conference on*, pages 188–195. IEEE, 2013. 1, 8, 33

[38] Rahul Venkataramani, Atul Gupta, Allahbaksh Asadullah, Basavaraju Muddu, and Vasudev Bhat. Discovery of technical expertise from open source code repositories. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 97–98. International World Wide Web Conferences Steering Committee, 2013. 1, 2, 34, 35, 54

[39] Gaj Vidmar and Nino Rode. Visualising concordance. *Computational Statistics*, 22(4):499–509, 2007. 17

[40] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13, 2008. 21

[41] Yiming Yang. Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 13–22. Springer-Verlag New York, Inc., 1994. 23, 35

[42] Jun Zhang, Mark S Ackerman, and Lada Adamic. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*, pages 221–230. ACM, 2007. 9, 11, 34, 35, 54

Appendix A

User mappings - Google Groups

ID	Name

Table A.1: User mappings - Google Groups

Appendix B

Related work

B.1 Expert identification

In addition to the related work discussed in Chapter 5 there are many more relevant papers on expert identification. Starting with a short database search the list of relevant papers was further expanded using the snowballing approach. Below follows a brief overview of those works with a description of the techniques used and the community the technique was applied on.

Paper	Method	Community	Comment
Pal et al. [26]	Behaviour	TurboTax Live	
Pal et al. [28]	Behaviour	TurboTax Live	
Venkataramani et al. [38]	Activity: Source code	GitHub, Stack Overflow	Uses data mined from GitHub to recommend experts on Stack Overflow.
Teyton et al. [35]	Activity: Source code	GitHub	Focusses on library usage.
Pratkanis [29]	Activity: Text, Social	Robot Operating System	
Guo et al. [14]	Activity: Text	Yahoo	
Alonso et al. [3]	Activity: Source code/tree, Social	Apache HTTP Web server project	Expertise is defined by user activity in specific parts of source tree.
Macek et al. [21]	Activity: CVS logs, Social	Conferator project	Uses RFID tags to monitor interaction between developers.
Zhang et al. [42]	Activity, Social	Sun Java forum	Multiple algorithms are introduced and implemented.
Kagdi et al. [16]	Activity: Source code	koffice, Apache httpd, and GNU gcc	Expertise is defined by looking at who changed a file.

Thongtanunam et al. [36]	Activity: Source code	Android Open Source Project (AOSP), OpenStack, Qt and LibreOffice projects	Experts are defined by looking at the file paths of the files they changed.
Anvik et al. [4]	Activity: Bug reports	Eclipse, Firefox	
Bouguessa et al. [8]	Activity, Social	Yahoo	
Adamic et al. [1]	Activity	Yahoo	Experts are defined by looking at the categories they participated in.
Tamrawi et al. [34]	Activity: Bug reports	Eclipse	
Anvik et al. [5]	Activity: Bug report	Eclipse	
Badashian et al. [6]	Activity	Stack Overflow, GitHub	

Table B.1: Related work - Expert identification

Appendix C

Additional results

C.1 Top 20 ggplot2 experts

Zscore	Zdegree	ExpertiseRank
736	474	736
474	736	474
270	270	270
195	824	195
824	195	824
1117	279	1117
279	1117	279
2107	209	2107
209	2107	209
1703	1703	439
1189	898	898
898	1189	1703
439	439	1025
1025	858	1189
1257*	1483	1383**
860*	210	210
1483	1257*	858
210	1025	860*
858	626*	1483
626*	1066**	1156**

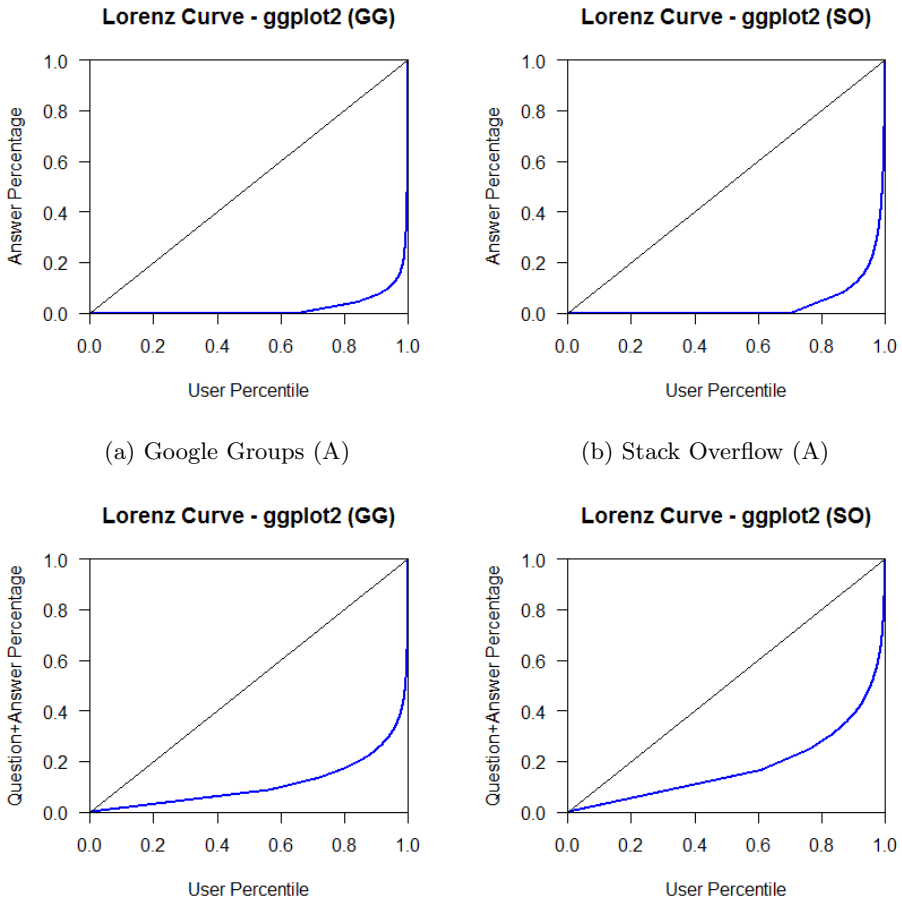
(a) Google Groups

Zscore	Zdegree	ExpertiseRank
1857266	1857266	1857266
324364	324364	324364
1838509	892313	471093
471093	471093	314020
892313	1838509	892313
602276	2985007	1412059
2985007	602276	602276
1855677	1855677	2372064
314020	1627235	1385941
1385941	1385941	2985007
1627235	2372064	1838509
2372064	1033808	2204410
1412059	1412059	1627235
1033808	314020	1855677
2204410	1851712	980833**
1851712	2204410	134830**
419994*	419994*	1033808
559784*	559784*	1851712
2725969*	2725969*	1900149**
415635*	190277**	415635*

(b) Stack Overflow

Table C.1: Top 20 ggplot2 experts in the Google Groups and Stack Overflow community. User Ids marked by * appear by 2 out of 3 algorithms and user Ids marked by ** appear only by 1 out of 3 algorithms.

C.2 Inequality Community Contribution



(a) Google Groups (A)

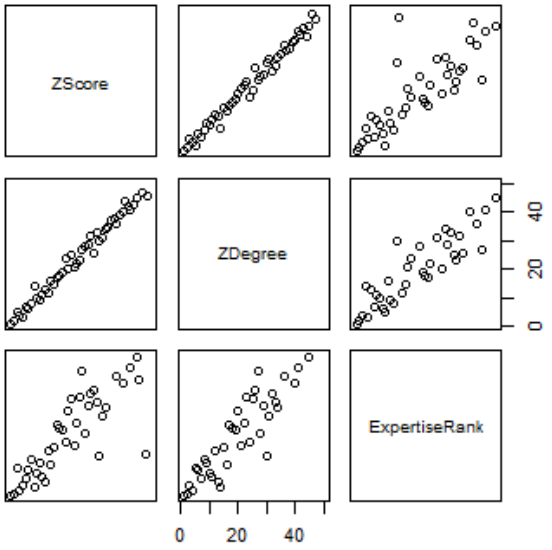
(b) Stack Overflow (A)

(c) Google Groups (Q&A)

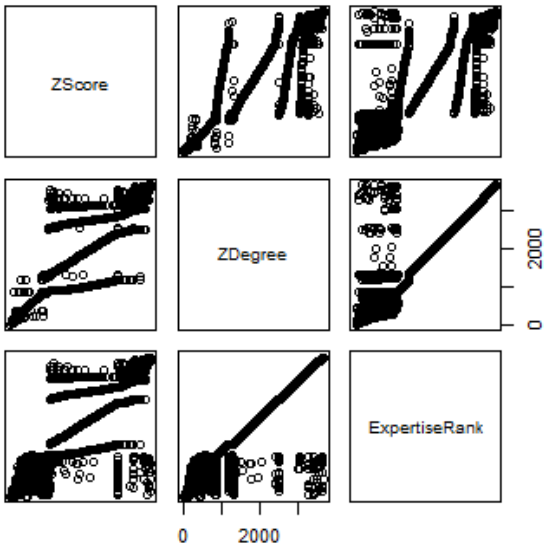
(d) Stack Overflow (Q&A)

Figure C.1: Lorenz curve of the ggplot2 community at Google Groups and Stack Overflow. Showing inequality for answer count and question&answer count.

C.3 General Expertise Algorithms



(a) Top 50 contributors



(b) All contributors

Figure C.2: Scatter plots showing the similarities in ranking between the different algorithms (Stack Overflow).

C.4 Focussed Expertise Analysis - Full dataset

In addition to the analysis in Chapter 4 where we applied 10-fold cross validation this appendix shows the results when using the complete dataset for training. When expert finding is being applied in practice the complete dataset set is being used as well, therefore the following results might be of more interest from a practical point of view.

The comparisons between the different approaches as discussed in Section 4.1 are shown in Table C.2 and Table C.3 for TF-IDF and Custom respectively and are tested on the Google Groups data set. From the discussed approaches we select TF-IDF (without term frequency) and Custom (without irrelevant terms) to further analyse the ggplot2 datasets of Google Groups and Stack Overflow, Table C.4. A visualisation of the ranking results is shown in Figure C.3.

	Term frequency	No term frequency	No irrelevant terms
Rank 1-10	2052	602	0
Rank 11-20	1117	779	1
Rank 21-30	527	950	1
Rank 31-40	171	1039	0
Rank 41-50	54	551	0
Total analysed	3921 (91%)	3921 (91%)	2 (0%)
Total	4307	4307	4307

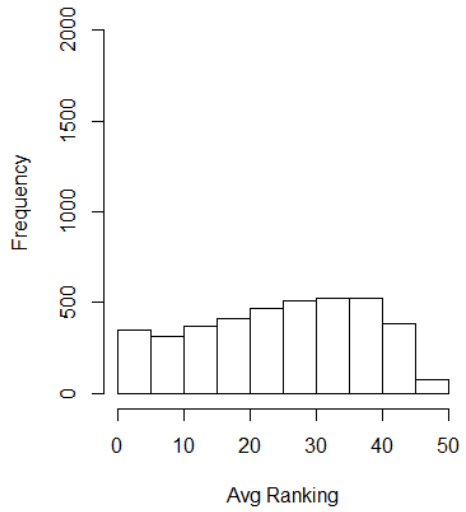
Table C.2: TF-IDF expertise ranking results (GG).

	Term frequency	No term frequency	No irrelevant terms
Rank 1-10	NA	134	2690
Rank 11-20	NA	225	630
Rank 21-30	NA	456	191
Rank 31-40	NA	913	54
Rank 41-50	NA	1863	18
Total analysed	NA	3591 (83%)	3583 (83%)
Total	NA	4307	4307

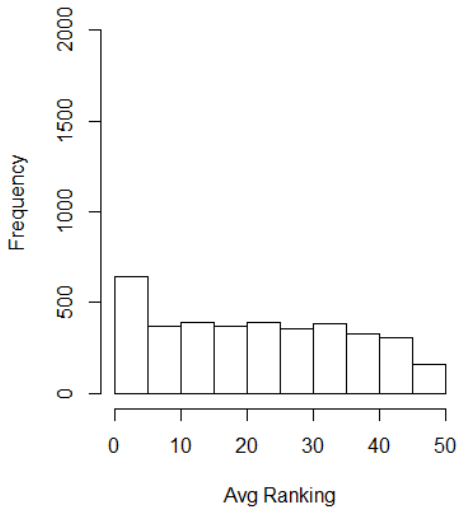
Table C.3: Custom expertise ranking results (GG).

	TF-IDF (GG)	Custom (GG)	TF-IDF (SO)	Custom (SO)
Rank 1-10	602	2690	938	1794
Rank 11-20	779	630	764	893
Rank 21-30	950	191	753	492
Rank 31-40	1039	54	730	232
Rank 41-50	551	18	511	132
Total analysed	3921 (91%)	3583 (83%)	3696 (66%)	3543 (64%)
Total	4307	4307	5563	5563

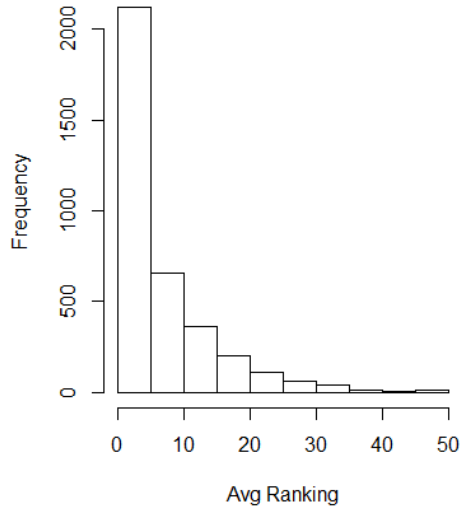
Table C.4: Expertise ranking results.



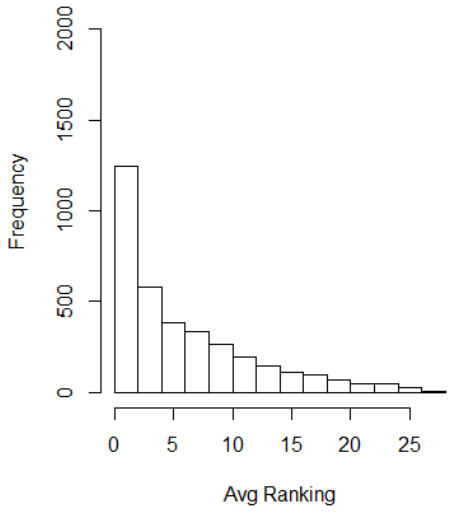
(a) TF-IDF (GG)



(b) TF-IDF (SO)



(c) Custom (GG)



(d) Custom (SO)

Figure C.3: Expertise ranking results.

Appendix D

ExpertiseRank Implementation

The code snippet below shows the ExpertiseRank implementation. The complete source code is available at the following GitHub repository: https://github.com/josdemmers

```
// Loops until change between current and last expertise rank is insignificant
while (!l_Converged)
{
    double l_Change = 0;
    // Foreach user active in the Google Groups group
    foreach (User l_UserX in m_Users)
    {
        l_UserX.LastExpertiseRank = l_UserX.ExpertiseRank;
        double l_Sum = 0;
        // Foreach user helped by x
        foreach (int l_UserY in l_UserX.InDegreeList)
        {
            // Expertise rank divided by the total amount of people who helped user y.
            l_Sum += m_Users[l_UserY].ExpertiseRank / (m_Users[l_UserY].OutDegreeList.
                Count);
        }
        // The summation of users helped by x as calculated above is multiplied by
            the damping factor. This value is added to the minimal ExpertiseRank (1.0
            - p)
        l_UserX.ExpertiseRank = (1.0 - p) + p * l_Sum;

        l_Change += Math.Abs(l_UserX.ExpertiseRank - l_UserX.LastExpertiseRank);
    }

    if (l_Change < .00001)
    {
        l_Converged = true;
    }
    l_Iteration++;
}
```

Listing D.1: C# implementation of ExpertiseRank

Appendix E

R Scripts

This appendix contains the R scripts that have been used to analyse and process the data from the data parsers.

```
#### GG ####
data1 <- read.csv("GG_stats_correlation.csv", header = TRUE, sep = ";")
data1 <- read.csv("GG_stats_correlation - Top50.csv", header = TRUE, sep = ";")

#### SO ####
data2 <- read.csv("SO_stats_correlation.csv", header = TRUE, sep = ";")
data2 <- read.csv("SO_stats_correlation - Top50.csv", header = TRUE, sep = ";")

data1$rank <- c(1:nrow(data1))#Add rank (Integer from 1 to size of data1)
all <- unique(sort(c(data1$ZScore,data1$ZDegree,data1$ExpertiseRank))) #list of all
    user ids; c combines data;
zscores <- c()
zdegrees <- c()
expertiseranks <- c()
for (user in all) {
  zscores <- c(zscores,as.numeric(subset(data1, ZScore==user)["rank"]))
  zdegrees <- c(zdegrees,as.numeric(subset(data1, ZDegree==user)["rank"]))
  expertiseranks <- c(expertiseranks,as.numeric(subset(data1, ExpertiseRank==user)[
      "rank"]))
}

#Indexes in df are sorted IDs, stats in other columns are the rankings of each
    users by an algorithm
df <- data.frame(all,zscores,zdegrees,expertiseranks)
names(df) <- c("UserID","RankAccZScore","RankAccZDegree","RankAccExpertiseRank")
names(df) <- c("User IDs","ZScore","ZDegree","ExpertiseRank")
#### Create a scatterplot
plot(df$RankAccZScore,df$RankAccZDegree)
plot(df$RankAccZScore,df$RankAccExpertiseRank)
plot(df$RankAccZDegree,df$RankAccExpertiseRank)
#### Create a mtrix plot of scatterplots
pairs(df)
#### Calculate the correlation coefficient
cor(df)#Default is spearman
cor(df, method = "spearman")
cor(df, method = "kendall")
```

Listing E.1: R script for measuring the correlating between ranking algorithms.

```
#install.packages("XML") #xmlParse
library(XML)
#install.packages("tm") #Corpus
library(tm)

##Google Groups##
doc <- xmlParse("GG_topics.xml", asText=FALSE)
messages <- xpathSApply(doc, "//Message//content", xmlValue)

##Stack Overflow
#doc <- xmlParse("SO_Posts_filter.xml", asText=FALSE)
#messages <- xpathSApply(doc, "//row", xmlGetAttr, 'Body')

#Make corpus for text mining
vector <- VectorSource(messages)
corpus <- Corpus(vector)

replaceExpressions <- function(x) UseMethod("replaceExpressions", x)
replaceExpressions.PlainTextDocument <- replaceExpressions.character  <- function(x
    ) {
  x <- gsub("[^[:alnum:]._]", " ", x) #Replaces all non alphabetic/numeric, dot,
      and underscore chars by space
  x <- gsub("\\d+\\.\\d+", " ", x) #Replaces doubles by space
  x <- gsub("\\.+[^a-z0-9]", " ", x) #Replaces dots at end of string by space.
  x <- gsub(" [[:punct:]]+", " ", x) #Replaces punctiation at start of string by
      space.
  x <- gsub("\\d+[a-z]+", " ", x) #Replaces string containing numbers follow by
      letters by space.
  x <- gsub(" \\d+", " ", x) #Replaces numeric string by space, i.e., 10 is
      replaced by log10 is not.

  return(x)
}

#TF-IDF
corpus <- tm_map(corpus, tolower)
#corpus <- tm_map(corpus, removePunctuation, preserve_intra_word_dashes = TRUE)
#corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, replaceExpressions)
corpus <- tm_map(corpus, replaceExpressions)#Repeat to replace remaining digits.
#corpus <- tm_map(corpus, stemDocument, language="english")
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, removeWords, stopwords("SMART"))
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, PlainTextDocument)
terms <-DocumentTermMatrix(corpus, control = list(weighting = function(x)
    weightTfIdf(x, normalize = FALSE)))

##Google Groups##
#matrix <- inspect(terms)
#Due to memory limitations tfidf terms need to be exported per 1000
matrix <- inspect(terms[18001:18341,1:39003])
df <- as.data.frame(matrix, stringsAsFactors = FALSE)
write.csv(df, file="TF-IDF_GG\\GG_wordlist_tfidf18001.csv", row.names = FALSE)

##Stack Overflow
#matrix <- inspect(terms)
#matrix <- inspect(terms[1:1000,1:48444])
#df <- as.data.frame(matrix, stringsAsFactors = FALSE)
#write.csv(df, file="TF-IDF_SO\\SO_wordlist_tfidf00001.csv", row.names = FALSE)
```

Listing E.2: R script for TF-IDF terms used in the focussed expertise analysis.

```
# load packages
#install.packages("RCurl")
library(RCurl)
library(XML)

# download html
html <- getURL("http://docs.ggplot2.org/current/", followlocation = TRUE)

# parse html
doc = htmlParse(html, asText=TRUE)
wordlist <- xpathSApply(doc, "//code//a", xmlValue)
wordlist
transform(wordlist)
write.csv(wordlist, file="GG_wordlist.csv", row.names = FALSE, fileEncoding = "UTF
    -8");
```

Listing E.3: R script for exporting all ggplot2 function names from the online documentation.

```
#install.packages("XML") #xmlParse
library(XML)
#install.packages("tm") #Corpus
library(tm)

##Google Groups##
doc <- xmlParse("GG_topics.xml", asText=FALSE)
userlist <- xpathSApply(doc, "//Message/User/Name", xmlValue)
topicidlist <- xpathSApply(doc, "//Topic/ID", xmlValue)
questions <- xpathSApply(doc, "//Messages//Message[1]//content", xmlValue)

##Stack Overflow
#doc <- xmlParse("SO_Posts_filter.xml", asText=FALSE)
#userlist <- xpathSApply(doc, "//row", xmlGetAttr, 'OwnerUserId')
#topicidlist <- xpathSApply(doc, "//row[@PostTypeId='1']", xmlGetAttr, 'Id')
#questions <- xpathSApply(doc, "//row[@PostTypeId='1']", xmlGetAttr, 'Body')

##GitHub##
#doc <- xmlParse("GH_posts.xml", asText=FALSE)
#userlist <- xpathSApply(doc, "//Message", xmlGetAttr, "UserId")
#topicidlist <- xpathSApply(doc, "//Topic", xmlGetAttr, "Id")
#questions <- xpathSApply(doc, "//Messages//Message[1]", xmlGetAttr, "Content")

#Make corpus for text mining
vector <- VectorSource(questions)
corpus <- Corpus(vector)

# process text
replaceExpressions <- function(x) UseMethod("replaceExpressions", x)

replaceExpressions.PlainTextDocument <- replaceExpressions.character  <- function(x
    ) {
  x <- gsub("[^[:alnum:].-]", " ", x) #Replaces all non alphabetic/numeric, dot,
      and underscore chars by space
  x <- gsub("\\d+\\.\\d+", " ", x) #Replaces doubles by space
  x <- gsub("\\.+[^a-z0-9]", " ", x) #Replaces dots at end of string by space.
  x <- gsub(" [[:punct:]]+", " ", x) #Replaces punctiation at start of string by
      space.
  x <- gsub("\\d+[a-z]+", " ", x) #Replaces string containing numbers follow by
      letters by space.
  x <- gsub(" \\d+", " ", x) #Replaces numeric string by space, i.e., 10 is
      replaced by log10 is not.

  return(x)
}

a <- tm_map(corpus, tolower)
#a <- tm_map(a, stemDocument, language="english")
a <- tm_map(a, removeWords, stopwords("english"))
a <- tm_map(a, removeWords, stopwords("SMART"))
```

```
#a <- tm_map(a, removePunctuation, preserve_intra_word_dashes = TRUE)
a <- tm_map(a, replaceExpressions)
a <- tm_map(a, replaceExpressions)
#a <- tm_map(a, removeNumbers)
a <- tm_map(a, stripWhitespace)

#a <- tm_map(a, PlainTextDocument)
m = as.matrix(t(a))

##Google Groups##
#Manually check output to make sure each question is on a separate line
write.csv(m, file="GG_questionlist.csv", row.names = FALSE);
write.csv(topicidlist, file="GG_topicIDlist.csv", row.names = FALSE);
write.csv(userlist, file="GG_userlist.csv", row.names = FALSE, fileEncoding = "UTF
    -8");

##Stack Overflow##
#Manually check output to make sure each question is on a separate line
#write.csv(m, file="SO_questionlist.csv", row.names = FALSE)
#write.csv(topicidlist, file="SO_topicIDlist.csv", row.names = FALSE)
#write.csv(userlist, file="SO_userlist.csv", row.names = FALSE, fileEncoding = "UTF
    -8")

##GitHub##
#Manually check output to make sure each question is on a separate line
#write.csv(m, file="GH_issuelist.csv", row.names = FALSE)
#write.csv(topicidlist, file="GH_topicIDlist.csv", row.names = FALSE)
#write.csv(userlist, file="GH_userlist.csv", row.names = FALSE, fileEncoding = "UTF
    -8")
```

Listing E.4: R script for cleaning / filtering question text.

```
library('ggplot2')

##TF-IDF##
df <- read.csv('GG_TFIDF_avg_rank_1.csv',sep=';',dec=',')

df$TFIDF_avg_rank
median(df$TFIDF_avg_rank)
mean(df$TFIDF_avg_rank)

TFIDF_avg_rank <- df$TFIDF_avg_rank
bins <- seq(-0.1,50.0,by=10)
ranks <- cut(TFIDF_avg_rank, bins)
table(ranks)
transform(table(ranks))

##Custom##
df2 <- read.csv('GG_Custom_avg_rank_1.csv',sep=';',dec=',')

df2$Custom_avg_rank
median(df2$Custom_avg_rank)
mean(df2$Custom_avg_rank)

Custom_avg_rank <- df2$Custom_avg_rank
bins2 <- seq(0.0,50.0,by=10)
ranks2 <- cut(Custom_avg_rank, bins2)
table(ranks2)
transform(table(ranks2))
```

Listing E.5: R script for analysing quality of the focussed expertise algorithms.

```
library('ggplot2')
library('reshape2')

#Google Groups
#Custom#

df2a <- read.csv('GG_Custom_avg_rank_1.csv',sep=';',dec=',')
df2b <- read.csv('GG_Custom_avg_rank_2.csv',sep=';',dec=',')
df2c <- read.csv('GG_Custom_avg_rank_3.csv',sep=';',dec=',')
df2d <- read.csv('GG_Custom_avg_rank_4.csv',sep=';',dec=',')
df2e <- read.csv('GG_Custom_avg_rank_5.csv',sep=';',dec=',')
df2f <- read.csv('GG_Custom_avg_rank_6.csv',sep=';',dec=',')
df2g <- read.csv('GG_Custom_avg_rank_7.csv',sep=';',dec=',')
df2h <- read.csv('GG_Custom_avg_rank_8.csv',sep=';',dec=',')
df2i <- read.csv('GG_Custom_avg_rank_9.csv',sep=';',dec=',')
df2j <- read.csv('GG_Custom_avg_rank_10.csv',sep=';',dec=',')

bucket<-list(a=df2a$Custom_avg_rank,b=df2b$Custom_avg_rank,c=df2c$Custom_avg_rank,d
    =df2d$Custom_avg_rank,e=df2e$Custom_avg_rank,
              f=df2f$Custom_avg_rank,g=df2g$Custom_avg_rank,h=df2h$Custom_avg_rank,i
                  =df2i$Custom_avg_rank,j=df2j$Custom_avg_rank)

# the melt command flattens the 'bucket' list into value/vectorname pairs
# the 2 columns are called 'value' and 'L1' by default
# 'fill' will color bars differently depending on L1 group
ggplot(melt(bucket), aes(value, fill = L1)) +
  geom_histogram(position = "stack", binwidth=2) +
  xlim(0,50) + ylim(0,1000) + xlab("Rank") + ylab("Count")
```

Listing E.6: R script for analysing quality of the focussed expertise algorithms.