Towards automated learning from software development issues

Analyzing open source project repositories using natural language processing and machine learning techniques

Author: Aleksandar Salov
Supervisor: Didac Gil de la Iglesia
Examiner: Ilir Jusufi
Exam date: 30 May 2017
Subject: Social Media and Web Technologies
Level: Master
Course code: 5ME11E
Abstract

This thesis presents an in-depth investigation on the subject of how natural language processing and machine learning techniques can be utilized in order to perform a comprehensive analysis of programming issues found in different open source project repositories hosted on GitHub. The research is focused on examining issues gathered from a number of JavaScript repositories based on their user generated textual description. The primary goal of the study is to explore how natural language processing and machine learning methods can facilitate the process of identifying and categorizing distinct issue types. Furthermore, the research goes one step further and investigates how these same techniques can support users in searching for potential solutions to these issues.

For this purpose, an initial proof-of-concept implementation is developed, which collects over 30 000 JavaScript issues from over 100 GitHub repositories. Then, the system extracts the titles of the issues, cleans and processes the data, before supplying it to an unsupervised clustering model which tries to uncover any discernible similarities and patterns within the examined dataset. What is more, the main system is supplemented by a dedicated web application prototype, which enables users to utilize the underlying machine learning model in order to find solutions to their programming related issues.

Furthermore, the developed implementation is meticulously evaluated through a number of measures. First of all, the trained clustering model is assessed by two independent groups of external reviewers - one group of fellow researchers and another group of practitioners in the software industry, so as to determine whether the resulting categories contain distinct types of issues. Moreover, in order to find out if the system can facilitate the search for issue solutions, the web application prototype is tested in a series of user sessions with participants who are not only representative of the main target group which can benefit most from such a system, but who also have a mixture of both practical and theoretical backgrounds.

The results of this research demonstrate that the proposed solution can effectively categorize issues according to their type, solely based on the user generated free-text title. This provides strong evidence that natural language processing and machine learning techniques can be utilized for analyzing issues and automating the overall learning process. However, the study was unable to conclusively determine whether these same methods can aid the search for issue solutions. Nevertheless, the thesis provides a detailed account of how this problem was addressed and can therefore serve as the basis for future research.

Keywords: machine learning, natural language processing, document clustering, issue categorization, issue classification, issue analysis, solution suggestions, open source, GitHub, project repositories
Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my supervisor Didac Gil de la Iglesia – I would not have been able to accomplish this without your invaluable guidance and expertise. Words cannot describe how thankful I am for your unwavering support not only during the last few months, but throughout the whole programme in general. Furthermore, to everyone who was involved in the evaluation process of my system – Maximilian Müller, Alisa Sotsenko, Janosch Zbick, David, Oscar, Ivan S., Ivan P., Mohamad, Abraham, Nurane, Todor and Svetlin, I want to express my gratitude for helping me thoroughly assess my prototype through your useful and honest feedback.

I want to thank everyone in the Department of Media Technology, with whom I have had the pleasure to interact with and learn from during the entirety of the programme. I also would like to thank all my classmates for helping create a pleasant and productive learning environment both inside and outside the classroom, and those of you in particular, with whom I have worked with on a number of projects and assignments.

Also, special thanks to all my friends for helping me keep my sanity during the last several months. Last but definitely not least, I want to thank my parents, for the constant emotional support that you have provided throughout my entire education and life as a whole.
# Table of Contents

List of Figures ........................................ VII
List of Tables ........................................... X
List of Code samples .................................... XII
List of Abbreviations ................................... XIII

1 Introduction ........................................ 1
   1.1 Problem domain ..................................... 2
   1.2 Motivation .......................................... 4
   1.3 Research questions ................................ 5
   1.4 Research scope ..................................... 7
   1.5 Contributions ...................................... 7
   1.6 Thesis outline ..................................... 8

2 Related work ......................................... 10
   2.1 Issue taxonomy ..................................... 10
   2.2 Preliminaries: NLP & ML .......................... 12
   2.3 Issue classification strategies .................... 14
   2.4 Document clustering ................................ 16
   2.5 GitHub issue analysis .............................. 17

3 Methodology ......................................... 20
   3.1 Goal setting ........................................ 21
   3.2 Literature review ................................... 21
   3.3 Conceptual design .................................. 21
   3.4 Implementation ..................................... 22
   3.5 Evaluation .......................................... 23
   3.6 Analysis ............................................ 23

4 Solution design ...................................... 25
   4.1 Design approach .................................... 25
   4.1.1 Target data ....................................... 25
   4.1.2 Data analysis ..................................... 26
   4.1.3 Data transformation .............................. 28
   4.2 Design phases ....................................... 29
   4.2.1 Issue categorization ............................. 30
# Table of Contents

4.2.2 Issue assignment ................................................. 32
4.2.3 Solution suggestions ............................................. 33

5 Implementation 35
    5.1 Issue categorization ............................................. 35
        5.1.1 Data collection ............................................ 35
        5.1.2 Data cleaning .............................................. 38
        5.1.3 Data preprocessing ........................................ 40
        5.1.4 Feature extraction ........................................ 43
        5.1.5 Feature clustering ........................................ 49
    5.2 Issue assignment ................................................ 53
        5.2.1 Data preparation ............................................ 53
        5.2.2 Classification ............................................. 54
    5.3 Solution suggestions ............................................ 55
        5.3.1 Category-based suggestions ................................ 55
        5.3.2 Similarity-based suggestions ............................... 63

6 System evaluation 67
    6.1 Issue categorization ............................................. 67
        6.1.1 Internal evaluation ........................................ 67
        6.1.2 Automatic cluster labeling ................................ 68
        6.1.3 Pilot testing .............................................. 73
        6.1.4 Expert analysis ............................................ 73
    6.2 Issue assignment ................................................ 76
        6.2.1 Evaluation approach ........................................ 77
        6.2.2 Data collection ............................................. 77
        6.2.3 Evaluation results ......................................... 79
    6.3 Solution suggestions ............................................ 80
        6.3.1 Pilot testing .............................................. 80
        6.3.2 User evaluation ............................................ 83

7 Discussion 88
    7.1 Implementation ................................................ 88
        7.1.1 Data ......................................................... 88
        7.1.2 Design approach ........................................... 89
    7.2 Results ........................................................ 91
        7.2.1 Expert analysis ............................................ 92
        7.2.2 Issue assignment ........................................... 93
        7.2.3 User evaluation ............................................ 94
    7.3 Limitations ..................................................... 96

IV
8 Conclusion 98
  8.1 Research questions .................................................. 98
  8.2 Contributions ....................................................... 100
  8.3 Future work .......................................................... 100

References 104

Appendix A Literature review protocol 111
  A.1 Research questions .................................................... 111
  A.2 Search scope .......................................................... 112
  A.3 Data items ............................................................. 113
  A.4 Search strategy ....................................................... 114

Appendix B Literature review results 116
  B.1 Resources found through literature search .......................... 116
  B.2 Additional relevant resources ...................................... 120

Appendix C Expert analysis protocol 123
  C.1 Research questions .................................................... 123
  C.2 Data selection .......................................................... 125
  C.3 Procedure ............................................................. 126
    C.3.1 Study design ....................................................... 126
    C.3.2 Participant roles ................................................ 126
    C.3.3 Study phases ..................................................... 127
  C.4 Limitations ............................................................ 129
    C.4.1 Data ................................................................. 129
    C.4.2 Study design ..................................................... 129
    C.4.3 Findings .......................................................... 130
  C.5 Discussion questions ................................................ 130
    C.5.1 Analysis ............................................................ 130
    C.5.2 Research .......................................................... 130
    C.5.3 Solution ............................................................ 131
    C.5.4 Evaluation ........................................................ 131

Appendix D Expert analysis results 132
  D.1 Individual labeling ................................................... 132
  D.2 Group review .......................................................... 134
  D.3 Post-review discussion .............................................. 136
    D.3.1 Analysis ............................................................ 136
    D.3.2 Research .......................................................... 137
    D.3.3 Solution ............................................................ 138
Table of Contents

D.3.4 Evaluation ................................................. 138
D.4 Follow-up analysis session ................................... 138
  D.4.1 Participant motivation .................................. 139
  D.4.2 Results ................................................ 140
  D.4.3 Analysis ............................................... 142

Appendix E User evaluation protocol .......................... 143
  E.1 Research questions ........................................ 143
  E.2 Procedure ................................................ 145
    E.2.1 Study design ........................................... 145
    E.2.2 Participant roles ...................................... 146
    E.2.3 Study phases .......................................... 146
  E.3 Limitations ................................................ 148
    E.3.1 Data .................................................... 148
    E.3.2 Participants ........................................... 149
    E.3.3 Study design .......................................... 149
    E.3.4 Findings .............................................. 149
  E.4 Questionnaires ............................................ 150
    E.4.1 Pre-test questionnaire ................................ 150
    E.4.2 Post-test questionnaire ................................ 151

Appendix F User evaluation results .......................... 152
  F.1 Pre-test questionnaire .................................... 152
  F.2 Prototype testing .......................................... 159
    F.2.1 Observations .......................................... 159
    F.2.2 User feedback ......................................... 160
  F.3 Post-test questionnaire ................................... 161

Appendix G Implementation code ............................ 168
List of Figures

3.1 A diagram illustrating the agile research methodology employed throughout this study ................................................................. 20

4.1 A diagram representing the full issue categorization workflow, resulting in the creation of a categorization model .................................................. 30

4.2 A diagram showing the issue assignment process, which utilizes the trained categorization model in order to assign a new issue to a given cluster .... 32

4.3 A diagram of the overall system workflow which serves to provide potential solutions to a given issue, via a web application interface ................. 33

4.4 A simplified visual representation of the hierarchical clustering tree, created by dividing the existing clusters into two additional sublevels ............. 34

5.1 Comparison of the difference in the clustering results between the K-Means and Mini-batch K-Means algorithms, when applied to the same dataset (Source: scikit-learn, n.d.) ......................................................... 49

5.2 A table showing all 6 GitHub “reaction” types (Source: GitHub Developer, n.d.) ......................................................................................... 57

5.3 An example usage of the GitHub “reactions”, which allow users to rate the quality of a given comment .............................................................. 58

5.4 A screenshot illustrating the visual appearance of the web application prototype in its default state (when the page loads) ................................. 59

5.5 A screenshot of the output (i.e. “solution suggestions”) produced by the web prototype as a result of a given search query ............................. 59

5.6 Two examples of Google’s rich answers shown in response to different search query types .............................................................................. 60

5.7 A screenshot highlighting the suggestion range slider, which specifies how closely related to the search query should the output of the application be . 61

5.8 A diagram illustrating all the levels of the clustering tree (marked in green), which are being considered for searching potential solutions, depending on the currently selected “precision” setting ........................... 62
5.9 A visual representation of the distance problem that may occur when a vec-
tor is located near the edge of its assigned cluster – there might be data
points placed in a different cluster that are closer in distance to it, than some
members of its own grouping .............................................. 64

5.10 A screenshot demonstrating the most similar issue suggestions offered by the
web prototype (shown below the main search results) .................... 66

D.1 A diagram showing an aggregation of the expert confidence levels regarding
the chosen cluster labels (the dark grey lines signify the standard deviation) 135

F.1 A diagram illustrating the self-proclaimed primary occupation of the chosen
participants ................................................................. 152

F.2 A chart showing the main areas of expertise (in relation to software develop-
ment) of the study subjects ............................................ 153

F.3 Charts demonstrating the relative JavaScript experience of the participants
as well as their most recent use of the technology ....................... 154

F.4 A chart showing the users’ self-assessed JavaScript knowledge level .... 155

F.5 An overview of the different online platforms and resources that users utilize
in order to search for solutions to programming issues ................. 155

F.6 A diagram illustrating how often the participants use various online resources
for finding solutions to programming-related problems ............... 156

F.7 A diagram demonstrating how helpful a number of online platforms are con-
sidered to be, according to the study participants ..................... 157

F.8 A chart showing the perceived ease of finding relevant information on several
online platforms, as indicated by the evaluation subjects ............. 158

F.9 A diagram illustrating the participants’ opinion regarding the helpfulness of
different answer types .................................................. 159

F.10 A diagram showing the perceived relevance of the search results produced by
the web application prototype ......................................... 162

F.11 A chart demonstrating whether users believe they were able to find answers
to their inquiries ......................................................... 162
F.12 A diagram illustrating the perceived ease of retrieving relevant information using the web prototype ....................................................... 163

F.13 A chart showing the perceived ability of the users to acquire new knowledge through the system ......................................................... 164

F.14 A diagram illustrating whether participants were able to discover any interesting information during their search .............................................. 164

F.15 A chart displaying how helpful the web prototype was considered to be, compared to traditional search methods normally used by the participants ......................................................... 165

F.16 A chart demonstrating the perceived usefulness of having an application dedicated to finding programming-related solutions (such as this web prototype) 165

F.17 Diagrams illustrating the perceived impact of the “precision” setting of the application as well as indicating which option (if any) produced the most useful results ......................................................... 166

F.18 A chart showing which part of the search results was considered to be most helpful by the study subjects ......................................................... 167
List of Tables

2.1 A description of the 8 defect type categories, part of the Orthogonal Defect Classification scheme (Source: Chillarege et al., 1992) .......................... 11

4.1 An example illustrating how textual data can be transformed into a numerical representation using the vector space model approach ............................ 29

5.1 A comparison of the different word transformations that occur when various verb forms go through the process of either stemming or lemmatization ............................ 42

5.2 An example illustrating the TF-IDF weight scores which may be given to some of the terms shared among several text documents ............................ 44

5.3 A sample of the terms excluded from the feature dataset because they appeared too often or too rarely .............................................................. 48

6.1 A sample of the terms contained within some of the resulting clusters, extracted using three separate approaches – differential, internal and random (the bold font indicates words gathered through multiple techniques) .................. 71

6.2 A sample of the full-length titles of the issues inside some of the clusters, extracted using an internal and a random approach (and separated from one another by short horizontal lines) ........................................... 72

6.3 A summary of the results acquired through several statistical measures, aimed at evaluating the issue assignment capabilities of the clustering model ............................. 79

6.4 A sample of the issue titles used for conducting the initial pilot testing of the web prototype ................................................................. 81

A.1 A summary of the different data items gathered as a result of the literature review, along with the specific reasons for their collection ........................................ 114

A.2 An overview of the various search keywords and phrases used to retrieve papers, which could help address the literature review research questions .................. 115

D.1 The individual label assignments of the analyzed clusters provided by the first group of expert reviewers, juxtaposed with the labels given during the pilot testing of the clustering data .................................................. 132
D.2 The final label assignments of the three experts (Group 1) given after the group review session, along with their confidence levels with regards to the suitability of the chosen labels ........................................ 134

D.3 The individual label assignments of the analyzed clusters provided by the second group of expert reviewers, juxtaposed with the labels given during the pilot testing of the clustering data .......................... 140

D.4 The final label assignments of the three experts (Group 2) given after the group review session, along with their confidence levels with regards to the suitability of the chosen labels ............................. 141
List of Code samples

2.1 A short snippet demonstrating an example of JavaScript code .................................. 15

G.1 Using the GitHub API to retrieve all project repositories that fit a set of predefined criteria .......................................................... 168

G.2 Collecting a sample of issues which meet the requirements of the research, from a given project repository .................................. 168

G.3 Filtering out all non-English issues from the collected dataset ......................... 169

G.4 An overview of the various text preprocessing activities involved in the preparation of the issue dataset ........................................ 170

G.5 Extracting the most relevant terms (i.e. features) from the document dataset 172

G.6 Grouping the text documents into separate clusters, using the K-Means algorithm .......................................................... 172

G.7 Reducing the dimensionality (i.e. the number of dimensions) of the feature vectors .......................................................... 172

G.8 Building the hierarchical clustering tree, by dividing the top level clusters into two additional sublevels .................................. 173

G.9 Assigning a new issue to the most suitable cluster/s, using the trained categorization model .......................................................... 174

G.10 Transforming new issues into vectors, through the use of the existing term vocabulary .......................................................... 175

G.11 Finding the most similar issues to a given entry, based on the vector distance 175

G.12 Retrieving the most highly rated comments from the set of most similar issues, identified within the assigned category ......................... 176
# List of Abbreviations

API | application programming interface  
---|-------------------------------------
DOM | document object model  
GUI | graphical user interface  
IoT | Internet of Things  
JS | JavaScript  
JSON | JavaScript object notation  
LSA/LSI | latent semantic analysis/indexing  
MDN | Mozilla developer network  
ML | machine learning  
NLP | natural language processing  
NPM | Node package manager  
ODC | Orthogonal Defect Classification  
PaaS | platform-as-a-service  
POS | part-of-speech  
Q&A | questions and answers  
SEO | search engine optimization  
SO | StackOverflow  
SVD | singular value decomposition  
TF-IDF | term frequency - inverse document frequency  
UI | user interface  
URI | uniform resource identifier  
URL | uniform resource locator  
VSM | vector space model
1 Introduction

The open source software development philosophy lies at the heart of technological innovation and its importance will only increase in the near future. Even though, the term “open source” does not have a formal definition, it generally refers to software which is “publicly available as source code and may be freely used, modified, and redistributed, without charge for the license” (Anthes, 2016). In recent years, the exponential growth of code hosting platforms like GitHub, SourceForge, BitBucket and so on, have allowed developers to easily share knowledge and collaborate with each other in order to create better software. As a result, these platforms, and GitHub in particular, which currently hosts over \(60^1\) million projects, have become a tremendous source of information that provide an abundance of insight into the inner workings of an open source project. That is why, the platform has become a very attractive target for many researchers in recent years, as mentioned by Kalliamvakou et al. (2014). However, so far there have been only a few studies that have focused on the issues that are posted there, which represent not only a fascinating study subject but also an untapped source of knowledge and insight about the various programming problems that occur during the development process. Therefore, it has become imperative for this platform to be thoroughly examined in order to gain a more in-depth understanding, which will not only help maintain and improve existing projects, but will also serve to benefit future ones. For example, by analyzing the different types of issues that occur in a given project, it would be possible to identify common problem areas and get a more accurate sense of the main difficulties faced by the end users.

However, due to the fact that, as mentioned, there is an enormous amount of projects hosted on GitHub and each project has a dedicated issue section, effectively analyzing that data is very challenging. In fact, performing any kind of adequate examination through the use of conventional means is virtually impossible. This is a problem that is becoming more and more severe in today’s world and is largely referred to as the problem of “Big Data”. There are many different ways to deal with this issue, with the most obvious solution being to utilize the processing and computational power of machines and let them do all the work for us. Unfortunately, there is a large number of tasks that involve much more than merely processing a lot of information. Instead, they require much more sophisticated analysis and often times, a deep understanding of the data itself.

Nevertheless, there are a number of emerging technologies which can help address the aforementioned problem. Two of the most prominent options, especially when dealing with textual data, are natural language processing (NLP) and machine learning (ML). As the name suggests, natural language processing aims to transform human language, along with all of its little idiosyncrasies, into a form that can be understood by a computer. On the other hand, machine learning makes use of self-learning algorithms which can evolve over

1 \text{https://github.com/about}
time and become better at their job, the more data they have at their disposal. One of the most notable examples of the capabilities of these techniques is perhaps the IBM Watson project\(^2\). Watson is a supercomputer with enormous processing capacity that utilizes these methods, among others, in order to not only “understand” natural language but also apply that knowledge in various ways, such as analyzing medical records and financial trends or even winning a popular quiz game show\(^3\). Furthermore, in recent years, new smart digital assistants like Siri, Cortana, Alexa, etc., at the heart of which also lie these very same techniques, have become more and more commonplace and are used by millions of people all around the world.

Due to the considerable potential of these technologies, they have also grown in popularity within the scientific community and have been used for a wide range of different purposes. This thesis makes no exception. That is why, the main goal of this research is to explore the analytical potential of natural language processing and machine learning techniques when applied to open source project issues. The thesis provides a detailed account of the research efforts performed in order to examine how these technique can be harnessed so as to facilitate such analysis and produce actionable insights that can help identify distinct types of issues occurring within such projects, based on the terms used to describe them. This knowledge can, in turn, be utilized not only for maintaining and improving future software projects but also for automatically finding potential solutions to these problems, which would be of enormous benefit to the entire open source community, ranging from novice users to seasoned professionals.

### 1.1 Problem domain

First of all, since the research is focused on analyzing “issues”, it should be specified what is meant by that term. Unfortunately, different scientific communities often use different wording to describe the exact same concept. For example, the terms “issue” and “bug” are often used interchangeably to refer to the same thing, however in some respects the word “bug” has a more limited scope since it implies the existence of a software defect, while “issue” could signify any sort of problem that may be encountered. Moreover, Antoniol et al. (2008) found that, out of the 1800 posts they gathered from various bug tracking systems such as those for Mozilla and Eclipse, less than half referred to actual bugs, which illustrates that such repositories serve a much broader purpose than just being a chronological archive of the defects that occurred in a given system. Therefore, this thesis will almost exclusively make use of the more encompassing term “issue” in order to better communicate the fact that the research aims to examine a wider topic than just software defects.

That being said, the problem that this thesis aims to address has a number of chal-

\(^2\) [https://www.ibm.com/watson/](https://www.ibm.com/watson/)

1. Introduction

...challenging aspects which contribute to its overall complexity. For instance, one of the primary reasons for choosing to focus on GitHub, namely the vast amount of unstructured data that the platform has to offer, also poses a problem because, as mentioned, due to its sheer size, this data cannot be efficiently processed and analyzed using conventional methods and techniques. In that sense, the thesis tackles one of the most major topics currently being discussed and examined within the research community - the problem of “Big Data”. However, Big Data presents a significant challenge not only because of its size but also because of its “variety” (Agnellutti, 2014 p.5). In the case of GitHub, the information about the issues reported in each repository is mostly unstructured, thus making it difficult to classify and analyze in an efficient manner. As a result, extracting valuable insights, such as being able to identify common patterns and relationships within this data, becomes a complex task that does not have an obvious solution. In fact, this very analysis is the most challenging aspect of working with Big Data, even more so than collecting the information itself or making use of it.

Furthermore, in most big open source projects, there are a large number of contributors that work together to develop or improve a software product. Usually, those contributors are not collocated during the time of coding, reviews and so on. Therefore, throughout the development process, they require tools which can help them coordinate, communicate and discover meaningful information in the project and for the project. One such tool is GitHub, which is not only used for synchronizing the source code across software deployments, but also integrates mechanisms for issue reporting as well as for managing discussions and conversations along with those issues, in order to solve them. These tools are indispensable for improving each one of the issues found in the source code or other parts of the project and allow them to be tackled on an individual basis. However, a tool like GitHub fails to provide an overview of a project describing the typology of issues that are more commonly discussed, or to highlight parts of the project that may require further attention. Other project management tools try to fill this gap by providing statistics like time consumed per issue, dependencies in issues, roadmaps and so on. Examples of such tools include Redmine⁴, Jira⁵, HP Service Manager⁶ and so on. The main issue with these tools is that the information pertaining to the issues must be manually entered by the developers or managers, via standardized yet project specific tags, which is difficult to do in itself, but even more so for large software projects. This becomes even more problematic if a project has already been in development for months or even years before any project management tool has been adopted to coordinate the efforts of the team. Questions such as “has somebody fixed an issue like this one before?” or “which kind of issues occur most often?” are not easy to answer, particularly when there is no contextual issue information and metadata

---

⁴ [https://www.redmine.org/](https://www.redmine.org/)
⁵ [https://www.atlassian.com/software/jira](https://www.atlassian.com/software/jira)
that could help address these concerns.

Moreover, the issues that are reported in a code repository such as GitHub can have a very diverse nature and are not limited solely to bugs. Furthermore, the entries do not have a particularly strict structure and are often not categorized in any way. The categorization itself is achieved through the use of labels, which serve as a sort of tags that identify the nature of the issue, however, there is no universally agreed label taxonomy that could be used. Apart from a few labels that are available by default, each repository can define its own custom labels that fit with their specific purposes and worldview (Cabot et al., 2015). However, in most repositories, labels are rarely used, if at all, mainly due to the fact that they have to be assigned manually. Therefore, any analysis that is to be performed, would have to merely rely on natural language semantics, which is further exacerbated by the fact that there is little to no oversight or supervision of what is being posted (although that may differ depending on the project in question), meaning that the issues are of varying quality. This poses an additional challenge to the research since all of this available information will have to be filtered through so that only the parts that are useful and relevant for the purposes of the study are selected.

Last but not least, even though there is an abundance of information on the web, sometimes finding relevant solutions to a given programming issue is rather difficult. In fact, since there are so many different resources which could be used for discovering a suitable answer, this leads to an information overload, which makes it even more challenging to find an appropriate solution. That is why, one of the primary goals of this research is to explore a novel way to cope with all of these problems and thus contribute toward developing a system that could address them.

1.2 Motivation

There are several reasons why the thesis focuses on this particular topic. First and foremost, the appeal of open source software and the software development field as a whole has grown significantly in recent years and as a result, more and more people are constantly getting involved in the community and starting to create, innovate and collaborate among each other. As more software projects are being created, this inevitably leads to an increasing amount of bugs and issues being encountered. However, as mentioned, even though some issues occur quite often, sometimes finding a solution is not straightforward, despite the fact that there may be a number of such solutions that already exist. Therefore, investigating whether utilizing natural language processing and machine learning provides a viable approach for addressing this problem and facilitating the process of finding a suitable solution to a given issue, by being able to pinpoint common flaws in previous projects in order to learn from them, is a prospect worth investigating, and not only from a purely theoretical standpoint, since the knowledge gained from such research would have a number of practical implications as well, as illustrated in the previous section.
Furthermore, the topic is extremely relevant in today’s technological landscape since, as explained earlier, open source software is an essential part of it. In fact, as Anthes points out, the percentage of companies which utilize open source technologies have “almost doubled between 2010 and 2015, from 42% to 78%” (2016). What is more, the problems caused by Big Data are only going to become more pressing in the coming years (Agnellutti, 2014 p.2), especially with the constant technological advancement leading up to the “Internet of Things” (IoT), the growing popularity of various social networks and the enormous amounts of transactional data that is continuously being generated. Moreover, the ever increasing number of historical records that are being kept, only serves to exacerbate the issue even further.

Besides, apart from the fact that there has been little previous research focused on investigating the platform, there are several other reasons why GitHub is an interesting and suitable choice for being the subject of this study. First of all, the fact that it hosts over 60 million open source projects means that it provides an enormously large amount of data for analysis. Moreover, the data itself is very extensive since it does not pertain only to bugs but also code improvements, feature suggestions and so on. It is also very diverse, since GitHub hosts many different projects that utilize various programming languages and technologies. Furthermore, the fact that the platform has more than 22 million active users, demonstrates that it has indeed become the “de facto social coding platform” (Russell, 2013) and that a large community has been built around it - people with diverse backgrounds, skill levels, knowledge and areas of expertise. Lastly, due to the popularity of the site, any research contribution that is made can have a considerable effect and the acquired knowledge can help a large number of developers as a whole.

With all of the aforementioned considerations in mind, it becomes abundantly clear that analyzing GitHub issues can yield considerable benefits and can facilitate the development of robust, high quality software applications. Hence, this thesis strives to provide a contribution in that particular regard, through the insights obtained as a result of this study.

1.3 Research questions

As already mentioned, the main goal of this research is to develop a system that analyzes open source project repositories hosted on GitHub in order to identify common problem areas and potentially even solutions that could help address these problems. The proposed system would gather data about the issues reported in a number of repositories and, by utilizing natural language processing and machine learning techniques, would analyze the collected information and attempt to establish commonalities and connections (Dumais, 2004) among the entries within the examined dataset. Based on the chosen topic and the aim of the work, the main research question (RQ) can be stated as follows:
**RQ:** How can natural language processing and machine learning techniques help analyze issues gathered from open source project repositories hosted on GitHub based on the user generated description of each issue?

This research question can be further broken down into three more specific questions (RQx) each focused on a particular aspect of the overall topic, which is to be investigated in more detail:

**RQ1:** How can these techniques facilitate the process of identifying distinct issue categories?

This question aims to determine how natural language processing and machine learning can be utilized in order to separate distinct issues into self-contained groups based on the specific type of problem that they represent. Furthermore, it examines the exact techniques that could be adopted for achieving this very goal.

**RQ2:** Which natural language processing and machine learning techniques can be applied to automatically assign an issue to a particular category?

The second question is closely related to the first and serves as a logical continuation to it, because once a set of issues has been divided into separate categories, the next step would naturally be to add new issues to these categories, based on their similarity to the already existing entries. Therefore, this question is supposed to investigate if this can be achieved through the same techniques as the ones used for the categorization itself or if other measures have to be taken.

**RQ3:** Which computable approaches can be utilized to identify possible solutions that are relevant for the identified issue categories?

Finally, the last question goes one step further, attempting to uncover which natural language processing and machine learning methods can facilitate the process of finding potential solutions to these issues, once they have been separated accordingly. The rationale here is that if the specific category which an issue falls into is known, this could conceivably enable the automatic identification of possible solutions. Again, like RQ2, it aims to determine if the aforementioned goal can be accomplished using the same approach and the same techniques as before or if the overall strategy will have to be tailored to the task at hand.

By investigating these questions, the research aims to make a contribution towards better understanding how to build and maintain open source software projects, which would not only help improve existing projects but would also be very beneficial for future development.
1. Introduction

1.4 Research scope

Since the subject area that this study aims to examine is extremely broad and multifaceted, it is important to clearly define the scope of the research. This will not only ensure that the outcomes attained as a result of the research efforts can be assessed more accurately and objectively but also provide a justification for any potential exclusions that have been made.

That being said, the main objective of this study is to examine how natural language processing and machine learning can be used for analyzing GitHub issues, within the context of a specific scenario. The scenario itself is focused on utilizing the aforementioned techniques so as to separate distinct issue types into independent groups and possibly finding solutions to these issues based on their category. Therefore, topics which may be relevant but do not directly pertain to the matter at hand, will be addressed to some extent but will not be examined in excessive detail. Furthermore, due to the exploratory nature of this research, finding the most optimal configuration for the developed solution in terms of accuracy and performance is not of paramount importance and is undoubtedly an aspect which could be improved and build upon, possibly as part of subsequent scientific studies. What is more, even if the contributions that this research provides, do not address the problem in its entirety, they can nevertheless serve as the basis for future research efforts that lead to finding solutions to even more complex and challenging issues.

1.5 Contributions

The main research contributions that this thesis provides are as follows:

- Employing natural language processing and machine learning techniques in order to thoroughly examine a specific target, namely issues posted on GitHub, which has not been studied previously in such depth, using a similar approach or with the same intent.

- Demonstrating how these methods can be combined in a novel way, so as to process, analyze and categorize user generated, unstructured data created by the software development community, and thus contribute to the overall improvement of the field.

- Developing a functional implementation that serves as a tangible representation of these specific technologies and exemplifies how they can be incorporated in a real world context.

Even though, as shown in Chapter 2, there has been a lot of research in the fields of natural language processing and machine learning, so far when it comes to the software
field, the scientific focus has primarily been on open source bug repositories. Furthermore, despite the fact that as Kalliamvakou et al. (2014) indicate, in recent years GitHub has become a more popular target for research, there have still been only a handful of studies examining the platform, or even more specifically the issues posted there. Moreover, it does not seem that there has been a study with similar research aims as this one or adopting a similar methodological approach meaning that the findings acquired as a result of this study will serve to fill a knowledge gap within the field. What is more, despite the fact that this research is solely focused on GitHub, the proposed solution can apply to other platforms as well, at least on a conceptual level, even though the actual implementation might differ from one application to another. Besides, the findings of this thesis will also be relevant to closed projects as well, as evidenced by the study conducted by Jonsson et al. (2015), who applied machine learning techniques used for analyzing open source projects to proprietary software, achieving similar outcomes as a result.

Another novel contribution that this research makes is the fact that it provides an in-depth account of how natural language processing and machine learning can be combined together in order to analyze programming issues based solely on unstructured, free-text description provided by the end users. What is more, by thoroughly describing the specific approach utilized for achieving this task as well as the various challenges and obstacles encountered throughout the development process, it demonstrates how this particular problem can be tackled in an effective manner and how to avoid the various pitfalls along the way.

Finally, unlike many other studies in the field, one of the chief outcomes of this research is the implementation of a functional prototype which serves to demonstrate how natural language processing and machine learning techniques can be leveraged in a real world context. Furthermore, the suggested solution is also evaluated with actual users representative of the main target group, which further highlights the benefits of this approach as well as its potential applications.

1.6 Thesis outline

This thesis is divided into eight main chapters, including this one. Each chapter addresses a different aspect of the research and contains several sections and subsections. Overall, the thesis is structured as follows:

- **Chapter 2** provides an extensive overview of the related research that has been conducted in a number of scientific domains which are relevant to the subject matter of this thesis. It discusses the state-of-the-art within various disciplines related to the topic of this study and explains how these previous research efforts are pertinent to the work that is to be performed. Furthermore, the information acquired during this phase also serves to guide all design decisions that are made throughout the rest of the project.

- **Chapter 3** presents the methodological approach that has been employed and followed throughout the entirety of the research process. It outlines the different stages of the project
work as well as the specific tasks that are performed as part of each phase. What is more, it specifies the various scientific methods which are adopted during different stages of the study and provides a justification for their particular choice.

Chapter 4 describes the conceptual design of the proposed solution as well as the different components that serve as its building blocks. It outlines all stages involved in the development of the final prototype and explains the purpose of each phase as well as the way it relates to the next one that follows.

Chapter 5 presents an in-depth description of the implementation efforts that are executed in order to develop the proposed solution. It introduces each stage in the development process starting from the initial collection of data, through the building of a machine learning model, up until the creation of a final web prototype that serves as a practical representation of the underlying system.

Chapter 6 gives a detailed account of the different evaluation measures that are taken in order to assess the developed solution. Since there are many aspects of the research that require some sort of a validation, there are a number of evaluation procedures that are performed involving both internal and external methods, all of which are described in this chapter.

Chapter 7 contains a thorough discussion regarding the design of the study, the developed solution and the results obtained from the various user evaluation sessions. It provides a detailed explanation of the relation between the various design decisions that have been made throughout the development phase and the outcomes of the subsequent evaluation. It also discusses some of the inherent limitations of the study and their corresponding effects on the whole procedure.

Chapter 8 summarizes the work that has been performed and highlights the overall research contributions of this study in relation to the scientific field and more specifically the fields of natural language processing, machine learning and issue analysis. What is more, it also describes various potential directions for future research as well as additional improvements that could be implemented in order to build on the work presented in this thesis.

Finally, the last part of this thesis is the Appendix which contains detailed information regarding the procedures that were followed throughout the research as well as a comprehensive account of the results obtained from the different evaluation studies conducted as part of the project work.
2 Related work

The primary topics addressed in this thesis, lie at an intersection of several distinct disciplines. Therefore, it is important to examine each of these subject areas, determine what the state-of-the-art within them is and how it relates to the work that is being conducted here.

Since the goal of this research is to analyze issues in order to categorize them based on their text description, and potentially facilitate their resolution, Section 2.1 examines if there are any issue taxonomies that have been commonly adopted within the software development sphere, which could potentially guide the classification. Furthermore, due to the fact that the study intends to utilize natural language processing and machine learning for achieving its objectives, Section 2.2 presents a more detailed explanation of what these technologies are and how they have been used in various scientific fields. Moreover, Section 2.3 investigates previous research efforts in the area of issue classification and tries to establish the specific methods and techniques that have been adopted so as to achieve this. Besides, due to the fact that, as mentioned, the issue data that is being collected is in a textual form, it is also important to understand not only how to process text but also how to detect similarity among different entries so that they can be accurately separated into distinct groups. Therefore, Section 2.4 tackles the topic of text document clustering and the specific techniques used to achieve that. Finally, since the research efforts are focused on GitHub, Section 2.5 examines prior studies which have analyzed different aspects of the platform and in particular, the issues posted in the various project repositories hosted there.

2.1 Issue taxonomy

Software development is a very diverse field and as such, the people working within it face wholly different challenges depending on their chosen area of specialization. This might explain why the literature review conducted as part of this research indicated that there is no standard classification that has been accepted throughout the field. As Seaman et al. (2008) state, some taxonomies were developed for very specific purposes while others aimed to be more generic so that they can serve as supporting mechanisms during the development phase of a given project. For example, the classification proposed by the IEEE board (1993), aimed to “define a common vocabulary with which different people and organizations can communicate effectively about software anomalies”. On the other hand, a taxonomy such as the one put forth by Weber, Karger and Paradkar (2005) is, like many others within the field, solely focused on software security. Another common difference between classification schemes is their structure - some of them are flat while others have a hierarchical structure with several levels of categories and subcategories (Ploski et al., 2007). What is more, as Ploski et al. (2007) point out, a lot of the available taxonomies suffer from ambiguity i.e. it is rather difficult to make a clear distinction between some of the proposed categories.
Nevertheless, it appears that the most widely adopted categorization is the one proposed by Chillarege et al. (1992) and developed at IBM, called Orthogonal Defect Classification (ODC). According to Freimut, Denger and Ketterer (2005), who have themselves made use of it as part of their work, ODC has become a “quasi-standard in industry and research”. Since specifying the nature of a given software defect is often subjective, the ODC scheme employs an alternative approach by instead classifying the fix required to address the problem, thus decreasing if not eliminating, the factor of subjectivity. The rationale behind this scheme is that, from a developer’s perspective, it is easier to describe the type of corrective work that was conducted in order to repair the bug than trying to define the root cause of the issue itself. In this context, the term “orthogonal” implies that the different categories are independent and mutually exclusive, so any given issue can only fall into one of them. According to Chillarege et al. (1992), the ODC classification can be applied to problems that are found at any stage of the overall development process. The scheme itself consists of 8 distinct categories to which an issue can be assigned, illustrated in Table 2.1.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Missing or incorrect functionality</td>
</tr>
<tr>
<td>Assignment</td>
<td>Variable assignment or initialization</td>
</tr>
<tr>
<td>Checking</td>
<td>Faulty or missing validation</td>
</tr>
<tr>
<td>Interface</td>
<td>Errors in interacting with other</td>
</tr>
<tr>
<td>Time/serialization</td>
<td>Resource sharing/concurrency</td>
</tr>
<tr>
<td>Build/package/merge</td>
<td>Errors in build process, package management</td>
</tr>
<tr>
<td>Documentation</td>
<td>Inaccuracies in documentation</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Efficiency or correctness problems</td>
</tr>
</tbody>
</table>

As shown, the categories are quite general and can be applied in a variety of situations and development scenarios. However, Ploski et al. (2007) argue that ODC, like most taxonomies found in the software development field, also suffers from the problem of ambiguity, at least to some extent. This claim is further supported by the work of El Emam and Wieczorek (1998), who used a classification scheme that was largely based on ODC and found that developers had difficulty distinguishing between some of the categories. Nevertheless, they also demonstrated that their scheme had a high degree of repeatability when applied to different defect datasets. Moreover, as mentioned by Seaman et al. (2008) and illustrated by their own work as well as the work of Freimut, Denger and Ketterer (2005) among many others, since its inception, the ODC categorization has been directly used or extended in a
number of research studies which serves as strong evidence of its value.

However, the biggest advantage of the ODC taxonomy might also be its most significant flaw in this particular situation. Due to the fact that its primary purpose is to be used ad hoc, assigning a category to a software defect once it has been solved, it would not be suitable for the purposes of this research. The reason for this is that in order to assign a label to a given issue based on the ODC classification, the solution to the problem has to be known. Unfortunately, for the most part, it is not possible to determine what the solution to an issue is, based on the user generated description of it. Furthermore, since, as mentioned in Section 1.1 the “Issue” section in most GitHub repositories contains more than mere defects, many entries will not be able to fit into either category, thus making the classification unfit for this specific case. Therefore, despite its various qualities and wide acceptance within the scientific community, the decision was made for this thesis not to utilize the ODC scheme, or any other issue taxonomy and instead base the categorization on the processed data extracted from the issues. Nevertheless, the taxonomy review showed that most classifications use between 8-12 categories, thus indicating that there is a certain level of agreement within the field regarding the general number of categories that issues can fall into. This insight can be very useful at a later stage in the study and can help determine the exact number of categories that should be formed as a result of the data analysis.

2.2 Preliminaries: NLP & ML

According to Liddy (2001), there no single definition of the term “natural language processing” (NLP), also known as “text mining”, that has been agreed upon among scholars. However, the concept itself refers to the use of different computational techniques that try to analyze human language in order to “understand” its meaning, or as Liddy calls it “accomplish human-like language processing” (2001). There are many different approaches that can be employed for performing such analysis, with one of the most common being the statistical method. As the name suggests, the statistical technique relies on mathematical formulas as well as “large bodies of linguistic data” (Bird, Klein and Loper, 2009) called corpora which serve as training data that can be used to create “approximate generalized models of linguistic phenomena” (Liddy, 2001). This means that based on its analysis of a large amount of observable data, an NLP model can make inferences about human language. Furthermore, text corpora are also often supplemented by dictionaries which can provide detailed information about different words, thus facilitating morphological and lexical analysis which leads to a more complete “understanding” of language and its various little intricacies. Nowadays, natural language processing is used in many different scientific fields and for a wide range of purposes. One of the most common applications for NLP, which has become extremely popular in recent years, mainly due to the rise of social media platforms, is sentiment analysis, which is also known as “opinion mining” (Pak and
2. Related work

Paroubek, 2010). Regardless of the exact term that is being used to describe it, the goal of this approach is to analyze a piece of text and determine the sentiment behind it i.e. if it is positive, negative or neutral. As mentioned by Pak and Paroubek (2010), this information can be extremely valuable, for instance, for companies trying to determine how their new product is being received or for political figures who want to get a sense of the public opinion regarding a specific subject. Another example application of NLP, which is especially pertinent to this research, is the work of Ko, Myers, Chau (2006), who utilized this technique so as to examine a large set of bug report titles, with the aim to try and find any discernible patterns that could help to better understand how people describe their software related problems.

Unlike natural language processing, machine learning (ML), also commonly referred to as “data mining” or “knowledge discovery”, describes a much more specific process. In fact, the statistical methods used for analyzing linguistic information, mentioned in the previous paragraph, rely on machine learning to perform their functions. Therefore, it can be concluded that machine learning is one of the ways to do natural language processing. However, what is machine learning in itself? According to Alpaydin (2010, p.2), the term refers to the process through which, by analyzing vast amounts of data, a statistical model is created. This procedure is also sometimes called “model training” since it serves to “teach” the model using past data (i.e. “training data”) so that it can readily analyze new instances of the same type that it encounters in the future. The purpose of this model is to identify patterns within the examined data, which are difficult to detect using alternative methods, not only due to the large size of the input but also because they may be very subtle. Furthermore, once a model is created, it is able to evolve or “learn” (hence the name) and become more accurate as it examines more new data. Due to the fact that today’s technological world has posed many complex challenges in a number of scientific disciplines and because of the exorbitant amounts of data that are available, particularly online, machine learning has found a variety of applications in many different domains such as retail, finance and many more.

Overall, it can be said that these two technologies are very closely related, especially when dealing with text. For instance, Maalej and Nabil (2015) utilized a number of natural language processing and machine learning techniques in order to examine app reviews in Apple’s App Store and Google Play and classify them according to several attributes such as the rating that was given and the textual description which goes along with it. By adopting this approach, the researchers were able to categorize the reviews into 4 major types with a precision of up to 95%, which demonstrates the potential of these methods for accurately categorizing entries based on user generated unstructured text.
2.3 Issue classification strategies

As the literature review showed, there has been an abundance of research dedicated to the topic of issue and bug classification. However, as demonstrated by the paper of Jonsson et al. (2015) who performed an in-depth review of previously conducted studies that address this subject, the majority of the research efforts in the field have been focused on analyzing the bug repositories of open source projects such as Mozilla, Eclipse, etc., instead of code repositories like GitHub.

For example, Xuan et al. (2015) utilized natural language processing and text classification techniques in order to automate the process of “bug triaging” i.e. assigning a bug to a developer that could fix it. The approach that they adopted, took into consideration both the title (in their case called “summary”) and the description of the bug so as to make the prediction. Another study that uses these same techniques is the one done by Schugerl, Rilling and Charland (2008), who employ these methods to evaluate the quality of bug reports that are being submitted. The authors base their criteria of what constitutes a “good” bug report on a wealth of previous research efforts conducted on the topic and perform a fine-grained analysis of the user generated free-form text (using NLP) aimed at assessing each quality metric. As a result, they are able to reliably determine the quality of a given bug report, which could help identify poor reports as well as the users who often submit them. Furthermore, Chaturvedi and Singh (2012) used 5 different classification techniques so as to determine the severity of a given bug by analyzing the text summary and description of the entry, while Antoniol et al. (2008) utilized this same data in order to divide issues into two categories - bugs and non-bugs.

On the other hand, Zhou et al. (2016) chose to employ a slightly different approach, taking into account not only the unstructured user summary but also the structured categorical data signifying the type, priority and severity of the issue in order to classify reports as either bugs or non-bugs. However, as evidenced by the research of Herzig, Just and Zeller (2013), since the person who reports the issue specifies all the structured metadata, this often results in misclassification because, in a lot of cases, the original poster does not have the required expertise. This misclassification in turn serves to bias the results of the analysis. Nevertheless, when reporting an issue on GitHub, the only information that can be specified is in the form of free text which means that this problem is avoided altogether.

Another study worth mentioning is that of Zanetti et al. (2013), who decided to tackle the problem from a different perspective, by analyzing the social network of bug reporters in order to predict the quality of a given post and determine whether it refers to an actual bug in the system. According to their findings, the more involved and active users are in the collaboration network surrounding an open source project, the more likely it is that they submit bug reports which describe actual problems. On the other hand, users who are not as involved or familiar with the project, are more prone to submitting duplicate reports or incomplete/incorrect entries. Based on this information, the researchers were
able to predict if a post refers to a genuine bug with up to 90% precision. This can have considerable impact for large software projects, where the contributor base is quite big and not everyone knows one another.

Pan, Kim and Whitehead (2009) also took an alternative approach to bug classification, a “reverse engineering” of sorts, by instead examining a large amount of historical data in order to identify common code fixes that were applied so as to address different bugs within a given software system. The authors claim that by identifying common source code changes which were implemented, it is possible to automatically assign a category to the bugs that necessitated these changes. Moreover, they argue that the chief advantages of this classification method are the fact that it helps avoid the subjectivity which inevitably comes with human categorization and that it could be applied across a wide range of software projects. Even though their paper presents an interesting solution to the problem at hand, it is not particularly well-suited for the purposes of this research. First of all, this thesis aims to address the problem from a user’s perspective and not so much from that of a developer working on the source code of the project. Therefore, the subject of interest are not the bugs that appear in the code of a given application, but rather the issues that users and other developers have to deal with when using this code. Furthermore, source code would not lend well to being analyzed through natural language processing techniques, primarily because, despite some similarities (especially in some programming languages), it cannot be regarded as natural language. The snippet\(^7\) below presents an illustrative example of JavaScript code, for comparison.

```
Code sample 2.1: A short snippet demonstrating an example of JavaScript code
1 var myFunctionReference = function() { /* do stuff here */ }
2 element.attachEvent(‘onclick’, myFunctionReference);
3 element.addEventListener(‘click’, myFunctionReference, false);
```

Lastly, Thung, Lo and Jiang (2012) combined both methods - analyzing the textual bug description along with the specific code fix that was implemented in order to address the problem, and used that data to categorize issues into one of three possible groups - data and control flow, structural and non-functional defects. If a similar approach was employed in this study, it would be possible to alleviate the problem outlined in Section 2.1 and make use of the ODC classification, because by examining the code changes that were applied as a response to an issue, it would be possible to determine what the solution actually was. However, that would likely involve a lot of manual analysis, which is something that this research aims to avoid. Besides, examining the code changes inside the various repositories, or as they are known on GitHub - “commits”, is out of scope for this thesis.

Even though, the goals of these research efforts mostly differ from the aim of this study, they nevertheless serve to exemplify the analytical potential of natural language processing

\(^7\) Taken from: [http://stackoverflow.com/a/6348597](http://stackoverflow.com/a/6348597)
and machine learning as well as validate the applicability of these methods in the domain of bug/issue classification and analysis. Furthermore, all of these papers demonstrate that the problem that is being addressed here has already been extensively examined by other researchers, thus proving both its relevance and importance. However, as mentioned, these studies focus exclusively on bug repositories and do not examine code repos. One of the reasons for this phenomenon is that code repositories have only come to the forefront in recent years, mainly because some of them did not exist before that. As a result, there are not as many papers that have investigated this topic.

Furthermore, despite the fact that these studies can serve as a valuable source of information about the use of natural language processing and machine learning in the context of issue analysis, their findings have a rather limited applicability in regards to this study. First of all, as mentioned, they deal with bug repositories instead of code repos. One of the main differences between the two is that, unlike code repositories, bug repos have a single purpose which is to allow users to report bugs. As a result, each entry that is being added there has a much more defined and rigid structure which makes the data more homogeneous and thus somewhat easier to analyze, even though, as stated by Herzig, Just and Zeller (2013), there is a lot of noise, redundancy and misclassification within it. Moreover, all of the studies presented in this section relied on pre-labeled data, meaning that the issues which they used for analysis were manually examined beforehand and assigned a label which was then used to determine whether the subsequent classification was correct or not. However, as mentioned in Section 1.1, the issues found on GitHub are rarely, if ever, labeled in any way, and that is highly dependent on the project that is being examined. Therefore, due to the fact that this research cannot take advantage of such neatly labeled information, alternative analytical techniques will have to be adopted so as to meet the objectives of the study. Furthermore, as stated by Coates and Ng (2012), “training only from unlabeled data is much more useful in a scenario where we have limited labeled training data”, as is the case with GitHub issues. Last but not least, nowadays, learning from unstructured, unlabeled data is far more relevant, since the majority of the information that can be found on the web, which as mentioned previously constitutes a huge amount, is in such a format.

2.4 Document clustering

The term “document” may seem a little misleading at first, because it implies a rather large piece of text such as a news article or scientific paper. However, in the field of text mining, it is universally used to refer to virtually any textual data that is being analyzed, regardless of its length. Therefore, the term will also be used as such throughout the thesis. On the other hand, “clustering” refers to a specific type of machine learning algorithm, which is used for dividing data into separate self-contained groups or “clusters”. The goal of this process is to try and find natural groupings within the data (Andrews and Fox, 2007) based on the similarity of the items. This means that the algorithm aims to separate the data so
that the entries within one cluster are as similar to one another as possible, while at the same time being as dissimilar as possible from the entries inside any of the other clusters (Manning, Raghavan and Schütze, 2009 p.349). Unlike most machine learning processes, such as classification, clustering is an “unsupervised” task, meaning that it does not rely on pre-labeled data. This makes it much more challenging to determine how well the trained model has been able to perform its function, because there is no “ground truth” that can be used as a point of reference.

The results obtained from the literature review indicated that there has not been a lot of research in the area of document clustering related to issues and bugs. Instead, most research efforts seem to be focused on text classification. The most likely reason for this is the aforementioned problem of evaluating the outcome of a learning scenario without pre-labeled data, which is difficult to both find and produce. As a result, most of the research on this topic consists of highly technical works which discuss different techniques for improving the accuracy of the algorithm (Dhillon and Modha, 2001; Schütze and Silverstein, 1997) or comparing various measures used for establishing similarity among text documents (Huang, 2008; Zhao and Karypis, 2004). However, these studies have a largely theoretical focus, while, due to the purposes of this research, the practical applications of this approach are the primary point of interest.

Nevertheless, one paper that is extremely pertinent to the work being done here, is the study of Raja (2013) who utilized a number of NLP techniques so as to analyze the text description of bugs in order to predict how long it will take for them to be resolved. However, unlike the papers mentioned in Section 2.3, Raja made use of clustering instead of classification so as to divide the entries into separate groups signifying different defect types, which can facilitate bug triaging and possibly provide an indication of how serious a given issue is.

Other studies, such as the ones by Pappuswamy et al. (2005), Kyriakopoulou and Kalamboukis (2006) and Lin and Wu (2009), use document clustering as a preprocessing step to text classification, aiming to discover latent structure within the data that could enhance the accuracy of the trained model. Furthermore, by utilizing this approach, they are able to facilitate the labeling process, since they do not have to label each individual document, but instead only the groupings formed as a result of the clustering. In fact, in some scenarios, the most prominent terms within each cluster of documents can serve as labels, thus completely eliminating the need for any sort of manual analysis.

### 2.5 GitHub issue analysis

As mentioned earlier, in recent years GitHub has become a popular research target and as a result, there have been a number of studies analyzing the platform as well as the issues that can be found there. For instance, Bissyandé et al. (2013) conducted an in-depth examination of around 100 000 GitHub repositories in order to establish how they utilize
their dedicated issue tracking systems (i.e. the “Issues” section of the repo). Their study found that the repos which primarily utilize their trackers are those belonging to large open source projects, which is not that surprising since as Kalliamvakou et al. (2014) state, a large portion of the projects hosted on the platform are either personal or no longer active. Another interesting insight that the researchers were able to find was that many of the issues are in fact reported by people who are involved in the development of the project (Bissyandé et al., 2013), which means that for many teams, GitHub serves not only as a platform that allows them to get direct feedback from users, but also as a way to collaborate among each other during the coding process.

On the other hand, Badashian, Hindle and Stroulia (2016) took an interesting approach for facilitating the process of issue triaging on GitHub. Their strategy, which they refer to as “crowdsourced” or “social” triaging, consists of analyzing the title and full text description of a given issue and using the most relevant terms within these fields to find relevant contributions that the developers involved in the project made, on the popular Q&A website StackOverflow\(^8\) (SO), in order to determine who should be assigned to solve the issue. For example, if an issue contains the terms “angular” and “directive”, their system would use those terms to search the StackOverflow profiles of the developers working on the project and examine whether they have answered any questions that have been tagged using these terms. Finally, the system evaluates the quality of their contributions, primarily signified by the number of upvotes they have received and determines the person that is the most suitable for addressing the problem. The researchers claim that this approach allows them to objectively assess the expertise that a developer has when it comes to a particular topic and thus helps automate the process of bug triaging, while at the same time ensuring that the issue is assigned to someone who can actually solve it. Of course, the most obvious drawback of this technique is that not all developers use both platforms and the ones that do, may not necessarily have linked their profiles or have the same username or email on both sites. Moreover, it is not unreasonable to think that some people might be extremely knowledgeable about a topic, but have chosen not to get involved in the StackOverflow community. However, using this triaging approach, those developers will be wrongfully dismissed in favor of others, potentially less experienced programmers.

Another paper focused on GitHub issues is that of Izquierdo et al. (2015) who created a visualization tool called “GiLA” which analyzes the issues within a given repository and provides a detailed overview of the various labels that have been used throughout the project. The tool provides a plethora of statistics regarding the label usage, such as which labels are the most popular, which are normally used together, which specific users are mostly involved with them and so on. However, as mentioned in Section 1.1 due to the fact that label utilization varies widely across different repositories, the GiLA tool would have a rather limited application - in some repos which are very thorough in their handling of

\(^8\) [https://stackoverflow.com/](https://stackoverflow.com/)
issues, it might provide a lot of insight, but in many other cases it will not be particularly useful. Nevertheless, Cabot et al. (2015) build upon the work done by Izquierdo et al. (2015) through their study which is also focused on analyzing the use of issue labels throughout the platform. Their findings confirm the notion that the majority of repos do not take advantage of the labeling mechanism at all, and most of the ones that do, rely solely on the default labels provided by GitHub. This means that either these repos consider the default labels to be more than enough for their purposes or that they just do not bother trying to define custom ones. Furthermore, the data collected by the researchers also showed that projects which grow in size, start using labels less and less. The assumption here is that since the labels have to be manually assigned, doing so becomes an extremely time-consuming task as the project evolves. Therefore, such projects can benefit even more from an automated labeling facility, which is one potential application of a solution that can categorize issues based on their textual description, like the one that is meant to be developed as a part of this research.

However, despite the fact that these studies have in one way or another examined issues submitted to the GitHub platform, none of the papers mentioned in this section have done it with the same goals in mind or by utilizing a similar approach. Therefore, it can be concluded that there is a certain knowledge gap, when it comes to automatically categorizing GitHub issues that this thesis intends to fill.
3 Methodology

Since the software development field is a very fast-paced and constantly changing environment, the development and research methodologies employed within the area, need to be flexible enough so as to accommodate the work process. That is why, the so-called “agile” methodologies, such as Scrum\(^9\), have become the de facto standard in the field, especially in recent times, as technology has become more advanced and ubiquitous, and the software more complex. These methodologies emphasize flexibility, adaptability and iterative development, where the different stages involved in the process are continuously repeated and progress is made in small increments. This workflow allows projects to naturally evolve, since it does not impose strict rules regarding the specific order of the development activities, which enables going back and forth between the different phases.

Therefore, the methodological approach employed throughout this research also has an iterative nature, which offers a high level of flexibility and allows for revisiting different steps or decisions, as new information is uncovered or generated as a result of the study. Figure 3.1 presents a general overview of the different stages involved in the overall process. As the diagram shows, the work that was performed can be divided into two main iteration cycles, which followed the same general procedure, as will any future research efforts.

![Figure 3.1: A diagram illustrating the agile research methodology employed throughout this study](image)

Each cycle consisted of several phases aimed at tackling a particular aspect of the problem at hand. Of course, the iterative process was also incorporated at this lower level of the study design, which facilitated jumping between separate activities. For example, as illustrated in Figure 3.1 if a new and unexpected challenge was encountered during the implementation, an additional literature review session was conducted so as to determine how to best overcome this obstacle.

The following sections present a more detailed description of each stage involved in the research process, along with the specific activities that were performed as part of it. What

\(^9\) [https://www.scrum.org/resources/what-is-scrum](https://www.scrum.org/resources/what-is-scrum)
is more, each section specifies the purpose of its corresponding phase and explains how it contributes towards addressing the main research topic of this thesis.

3.1 Goal setting

Once the research topic of the thesis was chosen, the next logical step was to define the specific goals that the study aimed to achieve. This was done by formulating several research questions, shown in Section 1.3, which clearly state what is to be examined as part of the project activities and with what purpose. These questions served as a guide throughout the subsequent research efforts and helped ensure that the thesis work was focused on the main study topic and did not stray away from it.

3.2 Literature review

The first step of the research process was to perform an in-depth literature review in order to establish what the state-of-the-art in the field of interest is and thus gain a more accurate perspective of the work that has been done so far. The review adheres to the guidelines outlined by Kitchenham and Charters (2007) which were used to establish the exact review procedure (see Appendix A). Following a strict protocol during the literature review process has several benefits. First of all, it adds a certain structure to the search activities, thus ultimately facilitating the review process. Furthermore, a protocol helps avoid bias and increases the reproducibility of the review itself, since it provides detailed instructions on how to conduct the entire study procedure.

The review consists of three phases - planning, searching and reviewing the results. The planning stage includes defining the research questions which would guide the review process as well as specifying the overall strategy and scope of the search activities. The next step is to conduct the review itself by adhering to the predefined procedure and collecting a number of scientific publications. Finally, the last stage of the process involves analyzing the gathered papers in more depth in order to guarantee that they fulfill certain quality criteria, so that they could be considered as reliable resources as well as summarizing the learning outcomes obtained from the analysis. This information is of utmost importance for determining whether current studies are relevant for the work that is being carried out.

3.3 Conceptual design

After forming a solid theoretical foundation based on the information acquired during the literature review, the next stage of the research process was to develop a detailed conceptual design of the proposed solution, which describes both the general idea behind it as well as the particular design decisions that were made in order to address the problem at hand. Since, there were three specific research questions posed during the earlier phases of the research, the design of the solution which was supposed to help address them, can be divided into
three separate, yet closely related phases, which represent the procedure that was developed for solving a particular problem.

The first such phase aims to tackle RQ1 and examine how natural language processing and machine learning techniques can be utilized for categorizing issues based on their text summary. As shown in Chapter 2 there are a number of different approaches which could be adopted so as to solve this problem. Therefore, the solution makes use of a number of NLP preprocessing facilities in order to prepare the data, before submitting it to a machine learning model which groups the various entries based on their similarity. The second design phase addresses RQ2 and tries to help determine how to assign issues to an already existing category. The approach chosen for accomplishing this, is very similar to the one utilized for the previous task. However, since the learning model has already been created, newly submitted issues only have to go through the preprocessing phase and then be introduced to the model, which automatically assigns them to a given category. Lastly, the third phase intends to provide an answer to RQ3 by examining how the solution can be used to find relevant answers to programming related questions. For this purpose, the trained machine learning model is further improved and a web prototype is developed, which makes use of the capabilities of the model in order to find relevant solutions to different issues. For a more comprehensive description of the overall solution design and the development stages involved in its eventual implementation, see Chapter 4.

3.4 Implementation

As mentioned in Section 1.5, one of the main contributions of this thesis is the development of a functional implementation, which not only serves as a means to tackle the main problem investigated in this research, but also demonstrates the specific design approach that was chosen for creating the proposed solution. Furthermore, the implementation helps to more effectively communicate the idea and, by putting it into practice, illustrates that it is not purely theoretical. However, the solution described in this thesis only represents an initial proof-of-concept prototype that requires a lot of additional work until it can be regarded as complete. Nevertheless, due to the fact that the study has an exploratory character, this prototype was more than enough for its purposes.

While the conceptual design of the solution serves as a theoretical representation of the idea behind it, the implementation depicts the practical embodiment of that design. Therefore, the information regarding the three main components which make up the overall design that was presented in the previous section, fully applies here as well, meaning that there is no need to repeat it again. However, for full details about the implementation phase of the research, please refer to Chapter 5.
3.5 Evaluation

Once the functional implementation has been developed, the next stage in the study process is to evaluate it. Since there are three separate components of the prototype, that calls for several different evaluations. What is more, in order to obtain more reliable evaluation data, which accurately represents the current state of the proposed solution, each component is assessed using two different evaluation methods - internal and external.

The internal method consists of several validation measures which serve as a form of quality assurance that helps assess whether the developed solution meets a certain standard and if any aspects of it need further improvement. These validation measures involve a combination of purely computational techniques, which evaluate the solution based on a set of predefined heuristics as well as an initial results analysis (i.e. “pilot testing”) conducted by the author. However, due to the possibility of researcher bias, the outcomes of these evaluations have to be further corroborated, which is where the external evaluation comes in.

As the name suggests, this method is based on the feedback acquired from external sources, such as other researchers, practitioners in the software field or technology-oriented students. This not only ensures that the evaluation results are not as biased, since none of the participants are in any way involved in the project or affected by its outcome, but also provides a valuable outside perspective that may, for example, help discover shortcomings in the system that had not been previously considered. Furthermore, each evaluation session follows a strict protocol (see Appendix C and Appendix E), which specifies how the procedure should be conducted from start to finish, thus eliminating many outside factors that may affect the outcome of the session, while also making sure that the study can be properly replicated. What is more, the external evaluations primarily utilize qualitative data collection methods such as observation and interviews, which allows for gathering richer, more detailed data (Lazar, Feng and Hochheiser, 2010 p.178). Moreover, as Gray states (2004, p.214), such methods are very well suited for research studies which have an exploratory nature (such as this one). Last but not least, due to the depth of the acquired data, the evaluation can yield an abundance of valuable insight, even if it is conducted with a relatively small number of participants. Chapter 6 provides an in-depth account of the various evaluation activities that were performed, along with their corresponding outcomes.

3.6 Analysis

Finally, after evaluating the solution, the last stage of the research process consists of reflecting upon the evaluation results and the feedback obtained from the test sessions, and forming conclusions based on the data. This process allows for identifying possible design or implementation flaws and helps understand how the proposed solution can be improved - information which can be used to prepare for the next iteration in the development pro-
3. Methodology

However, since the evaluation mainly relies on qualitative data collection methods, the analysis of the results requires some degree of interpretation. This means that different researchers can make varying conclusions based on their own perspective and area of expertise, even if they are looking at the exact same data. Therefore, even though the specific reasons and arguments for making any given conclusion are exhaustively described throughout this thesis, none of the formed conclusions claim to be absolute. Furthermore, since the goal of this study is to simply explore the main research topic, there is no need for making any definitive assertions. Having said that, Chapter 7 presents a comprehensive discussion of the study results and the feedback acquired from the evaluation participants, while Chapter 8 outlines some directions for future improvements, which can serve to address the shortcomings of the developed solution and build upon the work presented in this thesis.
4 Solution design

As mentioned, the main goal of this thesis is to investigate how natural language processing and machine learning techniques can be utilized in order to analyze issues gathered from open source project repositories hosted on GitHub, based on the user generated description of each issue, and as a result acquire actionable insights. More specifically, the research focuses on exploring the analytical potential of the aforementioned techniques in facilitating the separation of issues into distinct groups according to their type, which in turn can help discover potential solutions to these issues. The main challenge in accomplishing this is to make sense of a vast amount of data which is, for the most part, unstructured and diverse in nature and being able to extract some useful insights from it. Besides, the users who report these issues are often less experienced and knowledgeable and as a result, they may be confused about the nature of the problem itself.

Nevertheless, as the literature review revealed, NLP and ML techniques have been commonly adopted for addressing this problem and have proven to be effective in doing so. For example, as mentioned in Section 2.2, natural language processing methods can facilitate the analysis of any type of textual input - be it a simple expression or a full book, thus helping deal with the unstructured nature of the data. Besides, Big Data analytics is an area where machine learning excels at (Ma, Zhang and Wang, 2014). Furthermore, by its very nature, a machine learning model constantly improves which ensures that, if the model has been properly trained, the more data that it processes, the more accurate the results ought be, meaning that it can turn one of the main problems caused by Big Data, namely “volume” (Agnellutti, 2014 p.5) into a strength.

However, apart from selecting the methods which are utilized to tackle the problem at hand, it is even more crucial to decide on a particular approach for accomplishing this task. This chapter presents an overview of the overall approach chosen for building the proposed solution and describes each of the different phases involved in the process.

4.1 Design approach

There are many different design choices that have to be made in order to create a solution capable of addressing the problem examined in this study. What is more, these decisions serve to shape the entire development workflow and can have a considerable effect on the research outcome. This section outlines the most crucial design choices which guided the thesis work and provides a reasoning for these selections.

4.1.1 Target data

Since the main subject of this study are issues found in various open source repositories hosted on GitHub, the first choice that needs to be made is which part of the data will be
used for the analysis. As mentioned earlier, unlike most bug repositories, the issues found on GitHub do not have any structured data associated with them that specifies the nature of the problem. Of course, there is a lot of contextual information available, such as when was the issue created, who posted it, how many comments are there and so on. However, since none of this data is relevant for the purposes of this research, the only useful information comes from the title and the body of the issue. Therefore, this study focuses entirely on the issue title, following the same approach as the one adopted by Zhou et al. (2016), who cite a number of other studies which demonstrate that the title (or in their case the “summary”) of an issue is a much more suitable target for analysis than its full-length description. The main reason for this is the fact that the title not only already contains the most important information about the issue itself but it is also very concise and straight to the point - there is no “noise” or unnecessary details such as stack traces, code samples and so on, often found in the issue body, that could serve to give unwarranted importance to ultimately inconsequential terms and phrases. What is more, Ko, Myers and Chau (2006), focused their research exclusively on the title of almost 200,000 bug reports and demonstrated that despite its short size, the title can provide a wealth of useful information about the underlying issue.

4.1.2 Data analysis

After choosing the data for the analysis, the next step is to decide exactly how it should be examined. Since one of the main goals of the study is to categorize issues according to their type, it is important to find an approach that can facilitate this process and, as the literature review illustrated, such a classification can be achieved through the use of machine learning. However, when it comes to machine learning, there are two types of learning scenarios which can be employed so as to address a given problem - supervised and unsupervised. In supervised learning, the correct assignments of the different items (the “ground truth”) are known prior to the learning process and the goal of the machine learning algorithm is to predict the assignment of the entries as accurately as possible. On the other hand, unsupervised learning has an exploratory character since no ground truth exists beforehand and so the algorithm tries to “find regularities in the input” (Alpaydin, 2010 p.11). As noted by Antoniol et al. (2008), the fact that the data does not have to be labeled, makes unsupervised learning techniques a very appealing alternative for large scale learning tasks (such as this one). However, as the researchers point out, the unsupervised learning results are often “difficult to interpret” (Antoniol et al., 2008) and thus identifying any discernible patterns within the data is a challenging task.

Moreover, even though unsupervised learning is a very daunting task in itself, the difficulties that this process poses are exacerbated even further when working with text documents. As mentioned, in unsupervised learning, the expected results are not known beforehand, which is why the process is often referred to as “knowledge discovery”. However, in most
situations, even though the final output is unknown, the key components of the data known as “features”, which are used to uncover the latent structure within the dataset, are. In the majority of learning scenarios, the exact number and values of the features are known in advance and thus can be readily presented to the machine learning algorithm. For example, let us say that a company wants to perform a segmentation of their client base according to the number of purchases they have made in the past and the amount of money they have spent so as to identify people who have bought a small number of items for large sums of money and are thus more likely to spend more on a single purchase, since this information could help the company target specific individuals with ads regarding their brand new product. So in order to do that, the company utilizes an unsupervised algorithm which separates their customer base into several groups. However, in this particular case, the features used for performing the segmentation, namely the number of purchases and the total amount of money spent, are well-known in advance and in fact, their selection serves to shape the insights which are to be gained from analyzing the end results. Unfortunately, this is not the case for learning scenarios involving text documents. When dealing with textual data, extra steps have to be undertaken in order to uncover the most relevant features (in this case terms) which are to be used for performing the machine learning task, which is the reason why the process necessitates that the data undergoes extensive preparation before it is introduced to the machine learning algorithm. These additional measures naturally increase the overall complexity of the task at hand and introduce a wide range of variables, which is what makes the entire procedure such a formidable endeavor. Nevertheless, since the data that needs to be analyzed, namely issues posted on GitHub, is largely not labeled in any way, utilizing the unsupervised machine learning approach would be the most appropriate choice. Therefore, despite the aforementioned difficulties, the proposed solution makes use of this method.

One of the most commonly used unsupervised learning techniques is called clustering, which as the name suggests, aims to “find clusters or groupings of input” (Alpaydin, 2010 p.11). Since this is precisely what the study aims to achieve, the developed system utilizes the K-Means algorithm, which is one of the most popular clustering algorithms due to its effectiveness and relative simplicity. What is more, according to Coates and Ng (2012), “K-means clustering can be used as a fast alternative training method”, which can be “easily implemented at large scale”, thus making it very suitable for the task at hand. The way K-Means works is by first taking the desired number of clusters that need to be made as input. Then, the algorithm randomly assigns several data points as cluster centers and through an iterative process tries to find the most suitable cluster for each data point, constantly updating the cluster centers and the number of points in each cluster. This whole process is repeated until there is no cluster change over multiple iterations or until the predefined maximum amount of iterations is reached. However, as mentioned, when doing unsupervised learning, it is unknown in advance how many clusters are actually needed.
Therefore, another decision that has to be made is to select the number of categories, which the issues will be divided into. Unfortunately, this is not a trivial task. Having too few categories would result in many different kinds of issues being bundled together. On the other hand, having too many would have a negative effect on the real world applicability of the solution, since it will be difficult for people to differentiate between so many groups. Again, due to the goal of the project, the classification accuracy is not crucial, but rather demonstrating that natural language processing and machine learning techniques can be applied in this manner and provide valuable output.

Nevertheless, as mentioned in Section 2.1, the most widely used issue taxonomies in the software field have between 8 to 12 categories. This means that the general consensus is that the number of generic issue categories is around 8-12, which provides an initial indication about the specific number that could be selected. However, as explained earlier, due to the considerable challenge posed by the complexity of the text document clustering task itself, allowing the clustering algorithm more room for inaccuracies would likely help attain more favorable results. What is more, this would increase the chance for the algorithm to correctly separate issues that differ from each other instead of bundling them together. Besides, this more fine-grained differentiation between the various issue types will serve to facilitate detecting any subtle/latent characteristics of the data which might otherwise go unnoticed. Naturally, as a result of this process, there might be several clusters that contain issues which fall into the same category, however, this would be even more likely to be discovered and so such clusters could easily be merged together. Therefore, in order to have an appropriate number of clusters that would be large enough so that there is a relatively well-defined separation among the items within them, while at the same time not having an overwhelming amount of clusters which would be difficult to analyze, the decision was made to create around two and a half times more clusters than the number of categories found in the most universally adopted issue taxonomies, which resulted in a total of 30 clusters.

4.1.3 Data transformation

Even though choosing to adopt an unsupervised learning approach through the use of K-Means clustering, appears to be the most advantageous decision given the data that needs to be analyzed and the specific goals of the research, it also introduces some additional challenges. Since the K-Means algorithm can only work with numerical input, the titles of the issues need to be transformed into a numerical representation, which can then be used for various learning tasks. The problem is that this conversion needs to occur without any loss of data, meaning that the new “version” of the title, needs to capture the essence of the underlying text.

The most widely used approach for achieving this is the “vector space model” (VSM), which represents a collection of text documents as “vectors in a common vector space”
4. Solution design

(Manning, Raghavan and Schütze, 2009 p.120). This common vector space refers to the fact that the resulting vectors, which can simply be viewed as arrays of numerical values, have the same size regardless of the actual length of the text documents themselves. In its most basic form, the vector space is built by first extracting all unique words from the document collection and assigning them an index. Then, each document within the dataset is represented as an array that indicates which of the extracted words it contains. If a word is not found in a document, this is signified with a 0 and if it is, with a 1. For instance, Table 4.1 shows how three short sentences can be transformed into vectors using the vector space model. Since there are 6 unique words in this example - “he”, “loves”, “likes”, “is”, “watching” and “football”, the resulting vectors have a length of 6 dimensions. It should be noted that the ordering of the word indices does not have any significance. Therefore the same sentences can be represented using vectors with a completely different order of values, as long as the indices correspond to the same word.

Table 4.1: An example illustrating how textual data can be transformed into a numerical representation using the vector space model approach

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Vector representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>He loves football</td>
<td>[1, 1, 0, 0, 0, 1]</td>
</tr>
<tr>
<td>He likes watching football</td>
<td>[1, 0, 1, 0, 1, 1]</td>
</tr>
<tr>
<td>He is watching football</td>
<td>[1, 0, 0, 1, 1, 1]</td>
</tr>
</tbody>
</table>

So based on the vector representation of these sentences, sentence 2 and 3 are more similar to each other than to sentence 1, since they have more words in common. This illustrates one of the main strengths of VSM, namely the fact that it facilitates the process of comparing different entries and estimating how similar or dissimilar they are. Of course, the inherent shortcoming of this approach is that if two documents do not share the same terms, they will not be seen as being related even if they are semantically similar. However, VSM has been found to work quite well in a number of different contexts and so it is worth investigating whether it can contribute to the issue categorization efforts. Furthermore, there are other more complex VSM implementations which try to assess the importance of a given word within both the document and the dataset as a whole, while also taking into account multi-word expressions. Last but not least, it is also possible to take some additional measures so as to address the aforementioned shortcomings of this approach. All of these techniques are thoroughly discussed in Section 5.1.4 and Section 5.1.5.

4.2 Design phases

As mentioned in Section 3.3, the design of the system can be divided into three main phases that aim to address the three research questions posed in Section 1.3. Figures 4.1-4.3 present an overview of the conceptual design of the entire solution which illustrates all the different
stages involved in the overall workflow of the proposed system. The diagrams illustrate the entire development workflow – from the initial data collection stage up until the final outcome. Every step produces some sort of an output which is subsequently used at the next one until the workflow is complete. This relative independence of the different components allows for a much higher degree of flexibility, because depending on the outcome, one can easily go back and forth between these components and make modifications as needed.

This section provides a summarized description of each design phase as well as the evaluation efforts which will be conducted in order to assess its outcome, while Chapter 5 outlines each implementation stage in more detail. It should be noted that these phases are not necessarily self-contained entities and some activities occur at several stages in the system workflow.

4.2.1 Issue categorization

Figure 4.1 presents an overview of the issue categorization workflow – from start to finish. The first stage of this workflow and the implementation process as a whole, is naturally the collection of issues. For this purpose, the system makes use of the GitHub API\[^{10}\] in order to gather a substantial dataset of issues, which will be utilized for the analysis. After the data has been retrieved, it goes through rigorous cleaning and preprocessing in order to prepare it for subsequent machine learning analysis. This process includes transforming the data into a more suitable format, removing portions of it which are not of interest and cleaning missing entries so as to guarantee the overall integrity of the dataset, which in turn will contribute to the effectiveness of the machine learning algorithms applied to it. Then, the feature extraction stage serves to prepare the data for text analysis by transforming the titles into a vector representation, as explained in Section 4.1.3. This process will help identify similarities between issues and thus enable grouping them into distinct categories. The similarities are established by finding the most common and relevant terms inside the dataset and examining the different term combinations contained

\[^{10}\]https://developer.github.com/v3/
within the issues. Finally, the dataset is supplied to a K-Means clustering algorithm which aims to separate similar documents into clusters in order to identify distinct categories of issues. As a result, a machine learning model is trained, which not only divides the existing issues into categories, but can also be used for assigning new issues to one of these groups.

However, once the clustering model has been trained, it has to undergo some sort of an evaluation in order to determine how well it has been able to categorize the issues. As mentioned in Section 3.5, the evaluation consists of both internal and external measures. The internal evaluation is done through the use of a special heuristic which attempts to estimate how separated the resulting clusters are, which is a sign that they are well-formed and therefore the items within them are quite distinguishable. This is followed by an initial analysis of the clustering results performed by the author, which is based on a subset of data extracted from each of the formed clusters. The goal of the analysis is to assess whether the categories have a clear theme and is done by examining the output and trying to assign a descriptive label to each group. Then, for the external part of the evaluation, the same extracted data is presented to two independent groups of reviewers, who also try to label the categories as part of an expert analysis. If the results of these sessions indicate that there is a consensus between the reviewers regarding the cluster labels, this would serve as strong evidence towards RQ1.
4. Solution design

4.2.2 Issue assignment

As mentioned in the previous section, once the clustering model has been trained, it can readily be used for assigning new issues to one of the already formed groups, as illustrated in Figure 4.2. However, before that can happen, any new issues have to undergo the same cleaning and preprocessing activities as the dataset used for the model training. Once that is done, the newly submitted issue is supplied to the clustering model which puts it into the category that contains the most similar items, resulting in an issue which has been assigned to a particular cluster.

However, even if the model has managed to separate the issue dataset into well-formed categories, there is no guarantee that it is able to correctly assign new issues to the appropriate cluster. Therefore, this particular aspect of the implementation also has to be evaluated. The assessment is done by gathering several small datasets of new issues from GitHub, labeling them using the categories identified during the expert analysis, supplying them to the clustering model and comparing the assignments given by the human judge (in this case the author) and the algorithm. In order to minimize bias, the comparison is done not only through the use of multiple datasets, but also through several computational measures which aim to determine the accuracy of the assignment and ensure that there is a certain pattern to it (i.e. it is not random). If the evaluation shows that the model can accurately assign issues to the categories they belong to, that would help address RQ2.

Figure 4.2: A diagram showing the issue assignment process, which utilizes the trained categorization model in order to assign a new issue to a given cluster.
4.2.3 Solution suggestions

As shown in Figure 3.1, the second iteration of the development process focuses on a different aspect of the problem, namely how to utilize the issue categorization in order to find relevant solutions to different issues. For this purpose, a dedicated web prototype is developed, which allows users to search for answers to various programming related inquiries. The prototype itself relies on the already established system workflow as well as the assignment capabilities of the clustering model. As shown in Figure 3.3, a new issue submitted through the web interface goes through the issue assignment phase outlined in the previous section, before being allocated to a given category. Then, the system searches for the most similar issues within the assign cluster and retrieves the comments posted in relation to these issues. After that, the comments are sorted according to their rating and the best rated ones are presented to the end user as possible solutions to the search query.

However, since the dataset is quite substantial and the clusters will most likely not be well-balanced, searching for similar issues within them will be a time and resource intensive process. Therefore, the clustering model is further enhanced by creating a hierarchical clustering structure, as illustrated in Figure 4.4. It is important to note that the diagram shows a simplified view of the hierarchical “tree” in order to better communicate the idea. In reality, the tree has a lot more branches, namely 30 at the top level, since as mentioned in Section 4.1.2 the initial training is done with 30 clusters, and two sublevels of 5 clusters each. Therefore, when a new issue needs to be categorized, first it is assigned to one of the clusters from the top level, then to a subcluster at the second level and finally a cluster at the third level. This serves to facilitate the search for similar issues since it will only be done at the lowest level, where there are not as many issues. The hierarchical structure itself is created in an iterative fashion by again utilizing the K-Means algorithm so as to divide the top level clusters into two further sublevels.

Finally, in order to assess whether the web prototype
can support the process of finding solutions to programming related issues, two separate evaluations are conducted. First, a pilot testing using randomly collected GitHub issue titles, again performed by the author. This is then followed by a number of user testing sessions done with people representative of the main target group that will benefit from such a solution to the greatest extent. If the user evaluations show that people find the solution to be useful and that it can help them in their search for issue solutions, this would provide a lot of evidence for answering RQ3.

Figure 4.4: A simplified visual representation of the hierarchical clustering tree, created by dividing the existing clusters into two additional sublevels
5 Implementation

After conceptualizing the potential solution that can help tackle the problem examined in this thesis, the next step is to put the idea into practice and develop a functional implementation based on it. As shown in Figures 4.1-4.3, the proposed solution incorporates several components designed in order to address each of the research questions, mentioned in Section 1.3. This chapter presents a detailed account of the development work done as part of the research efforts and provides an in-depth explanation of the different design decisions that were made throughout the process.

Nearly all of the code created as part of the development activities performed throughout this research was written in Python, which has become a popular choice among the scientific community for conducting large scale data analysis tasks due to its “strength in general purpose programming” (McKinney, 2012 p.2) as well as the extensive amount of dedicated third-party packages and libraries which have been specially designed so as to facilitate such tasks and can be readily included into any given project. Samples illustrating the most important parts of this code, can be found in Appendix G.

5.1 Issue categorization

As mentioned, one of the chief goals of the proposed solution is to analyze issues gathered from GitHub and group them according to their type. However, accomplishing this is by no means a trivial task and therefore, the categorization functionality that was implemented, consists of several stages which aim to prepare the raw textual input and transform it into a useful output. This section thoroughly describes each of these stages - from the initial data collection up to the final categorization of issues, focusing on their specific purpose within the overall design of the solution, and explains how these measures contribute towards meeting the main research objectives.

5.1.1 Data collection

Naturally, the first stage of the categorization process, and the solution implementation as a whole, is the data collection. As the name suggests, it consists of gathering relevant information that can be utilized for further analysis. This is achieved through the GitHub API which allows for collecting a lot of data about the various project repositories hosted on the platform. The API is freely available and provides access to both authenticated and unauthenticated users (naturally, different rate limits apply). In fact, as stated by Russell (2013), the GitHub API enables users to gather enough data that they could nearly recreate the experience offered by the platform itself. The API allows users to access the issues of each public repository that makes use of its dedicated issue tracker, meaning that an enormous amount of data can be extracted.
5. Implementation

However, GitHub considers issues and “pull requests”, which essentially represent code fixes that a contributor suggests to be applied, to be closely related entities. As a result, the API returns both entities as part of its response, whenever a user requests to access the issues of a given repository. The reason for this is that pull requests are often times a direct response to an issue that was posted i.e. a code change made in order to address the problem. Nevertheless, pull requests are not actual issues in themselves and as such are not relevant for this study and are therefore ignored.

Since, as mentioned, there are over 60 million projects hosted on the GitHub platform, it would not be possible to collect all the issue data that is available. Furthermore, due to the fact that the implementation will serve as a proof-of-concept prototype which aims to validate the chosen research approach, the exact number of issues gathered, even if it may influence the accuracy of the overall system, is not crucial. Moreover, collecting a more homogeneous, high quality data will serve to facilitate the automated analysis and thus increase the likelihood of finding distinct patterns within it. That is why, all project repositories that were being considered as well as the issues within them which were to be collected, had to fulfill a number of requirements.

First of all, the study focused on repositories based on a single programming language, namely JavaScript. The reason for choosing issues about a single language is that, since they all concern the same general topic, the data will be more cohesive, which as mentioned will facilitate the learning algorithms later on. In other words, this eliminates one of the independent variables that can affect the system, namely the coding language. Furthermore, it will avoid the problem of creating issue categories based on programming language, which is a highly likely scenario due to the fact that different languages make use of different terminology. Even though there is some merit in dividing issues according to such a criteria, this research aims to go deeper and separate issues in a more fine-grained fashion. On the other hand, the reason for choosing JavaScript is the fact that it has been consistently growing in popularity in recent years (StackOverflow, 2017) and as a result many new JavaScript libraries and frameworks are constantly being developed. This provides an abundance of data to work with, which tackles programming issues that are currently relevant for an extremely large user base.

What is more, the collection was only limited to the most popular JavaScript repos found on GitHub, since their popularity serves as proof that these projects are of high quality and have been well-received within the community. Both in GitHub and the open source field in general, there is a certain meritocracy and so good projects are bound to rise to the top, sooner or later. Furthermore, only considering repositories that have received a large amount of approval, helps avoid the majority of the “perils” outlined by Kalliamvakou et al. (2014), such as projects being inactive or personal, the latter of which is not a bad thing in itself, but if the project is not used by many people, there will be an extremely small number of issues to analyze (if any). This notion is further supported by the research of Bissyandé
et al. (2013), who found that the issue tracker natively provided by the GitHub platform is mainly used by large projects, which usually tend to be among the most highly rated. Last but not least, due to the popularity of these repositories, the issues being discussed there would be particularly relevant to the modern day software development field. On GitHub, the popularity of a given repo is estimated through the total number of “stars” that it has, which are similar to the “likes” on more traditional social media websites, since they allow developers to express their appreciation for a given project. Therefore, after some contemplation, the decision was made to only gather data from JavaScript repositories which have at least 10000 stars. This threshold both ensured that the repos meet a particular quality standard and that there is enough data to be collected.

This resulted in a total of 127 repositories, which would produce a dataset that is large enough to work with and provide useful results but not so big that it becomes an obstacle to the development process, due to the processing and memory limitations of the hardware used for developing the implementation. Furthermore, the GitHub API has a predefined request limit of 5000 requests an hour, meaning that even when working with a powerful machine, it is still not possible to go over the specified threshold (unless by using a distributed approach for this data collection, which is beyond the scope of this project), thus further restricting the amount of data that can be collected in a reasonable amount of time. The selected 127 repos represent the most influential JavaScript projects hosted on the platform, ranging from extensive frameworks and libraries developed by big corporations to small personal undertakings that have gained the favor of the community and have been recognized for their merits. However, all of these repos only make up a microscopic portion (0.004%) of the over 3.1 million JavaScript repositories residing on GitHub. This provides a real sense of the enormous scale of the data and perfectly illustrates one of the major challenges of working with Big Data.

So after deciding on the specific criteria used for the selection of eligible repositories, the next step was to determine the selection criteria for the issues themselves. The first such criterion for the issues that are being collected is that they have a status of “closed”, meaning that they have already been resolved - either because a solution was found or because it was not an actual issue in the first place. What is more, in case of the former, this would imply that a specific solution was provided in response to the problem, which is another aspect that is of interest to this research. Another limitation that was imposed was to only collect issues that have been posted since the beginning of 2015, which ensures that the data is not “stale” or no longer relevant. This is an important concern because, as evidenced by the study of Jonsson et al. (2015), the “freshness” of the data is more crucial than its sheer size, when it comes to improving the classification and prediction accuracy of a learning model. Lastly, in order to obtain a representative sample of the issues posted within each repo, the collection was done proportionately, meaning that the same portion, namely

\[ \text{https://api.github.com/search/repositories?q=language:javascript} \]

37
25% (i.e. \(\frac{1}{4}\)) of the total number of issues which fulfill the aforementioned criteria, were gathered. Moreover, all issues were selected at random, which serves to further decrease the detrimental effects of possible bias within the dataset. Furthermore, for each issue that is collected, all of the corresponding comments that were posted (if any) were also gathered, since they would be needed at a later point in the development process in order to facilitate the finding of possible solutions.

At the time of writing, the total number of closed JavaScript issues that have been created since the beginning of 2015 is over 2.4\(^{12}\) million. However, since the collection only considers repositories that have at least 10000 stars and gathers 25% of the total, the size of the resulting dataset ended up being over 30000 issues (31859 to be exact), which is around 1.25\% of all closed JavaScript issues on the platform. This serves to confirm the findings of Bissyandé et al. (2013) that a very small portion of the repositories hosted on the platform (the most popular and used ones) contain the large majority of issues. What is more, even though some of these repositories are not actual software projects but instead store different artifacts such as a JavaScript book series\(^{13}\) or a code style guideline\(^{14}\), the issues posted there are still pertinent to various development problems that users have encountered during their work.

5.1.2 Data cleaning

The next step in the implementation process was to “clean” the dataset (i.e. discard unwanted or missing data) in order to ensure that there are no entries which do not provide any useful information for analysis and thus could only serve to bias the final outcome. Furthermore, a clean dataset is of a better quality and has a greater overall consistency, which again can have a positive impact on the subsequent analytical efforts. Therefore, first of all, since the analysis of the collected issues is done based on their user generated text description, all entries that are partially or completely missing this information were removed from the dataset. Due to the fact that the GitHub API returns the issue data as two separate fields, namely “title” and “body”, if an entry that has been collected had a null value or an empty string for any of these fields, it was immediately discarded.

The next measure that was taken was to remove any duplicate entries. In this case, duplicate entries or “reposts” are considered only issues that have the exact same title and body. The reason for this is the fact that, for example, several issues having the same title but different body is a very likely scenario, since the title is, by its very nature, shorter in size. Furthermore, since the title has to describe the issue as well as possible, while abiding to a predefined length limit, there is a higher likelihood that it uses common keywords.

---

\(^{12}\) [https://api.github.com/search/issues?q=language:javascript+type:issue+state:closed+created:%3E=2015-01-01T00:00:00Z](https://api.github.com/search/issues?q=language:javascript+type:issue+state:closed+created:%3E=2015-01-01T00:00:00Z)

\(^{13}\) [https://github.com/getify/You-Dont-Know-JS](https://github.com/getify/You-Dont-Know-JS)

\(^{14}\) [https://github.com/airbnb/javascript](https://github.com/airbnb/javascript)
that will be repeated in a number of entries. Last but not least, such issues could serve to validate the approach of choosing to establish similarity using the title because if issues that share the same title but have a different body are in fact similar or related, they will rightfully end up in the same category. In such a case, the algorithm will put them together despite the fact that they could have been described using very different wording and so if the body of the issue was considered instead, they could have been wrongly separated. Of course, in order to make such a conclusion with a sufficient degree of certainty, the dataset has to be analyzed in depth, which is out of scope for this research. On the other hand, if some issues have a different title and the same body, which is a scenario that is much less likely to occur, it could again help validate the accuracy and reliability of the categorization process in case they are put together in the same group. Since such entries represent more or less the exact same issues, if they end up in the same category, this would demonstrate that the clustering algorithm is able to correctly determine if two issues are similar to each other.

However, even if some issues fit the aforementioned criteria, they would only be removed from the data, if there are no comments associated with them. The rationale for not removing all, is that even though such issues are obviously reposts, they nevertheless have the potential to contain different and useful comments, especially, considering the fact that they might have been posted in separate repositories, meaning that unless someone who follows closely several repos has seen the issue being posted somewhere else, people would be unaware that it is a repost and will most likely try to provide some sort of an answer. This could have important consequences for the second part of the implementation, which aims to find potential solutions to a given issue. Of course, these issues might also not be useful at all - for example, having a single answer which states the fact that it is a repost, however, finding an answer to this question would necessitate further research. On the other hand, issues that have the same title and body and contain no comments can be regarded as exact reposts that do not provide any extra information and therefore there is no actual value in keeping them. In this case, only the most recent instance of the post is kept, while the rest are removed from the dataset. A quick glance at the data shows that the decision to keep the duplicates with comments, resulted in a dataset of 31387 issues, instead of 31379 if they had been removed, meaning that the difference is quite small. Despite the fact that such entries could potentially have a slight negative effect on the accuracy of the clustering, the data shows that these occurrences are quite rare and can be considered as edge cases and so their impact is not particularly significant. Nevertheless, this is also something which requires further investigation.

Finally, the last step of the cleaning process involved filtering through all issues which were not written in English. The reason for this was the fact that the natural language processing techniques which are going to be applied to the data, cannot work on a multilingual dataset and so in order to achieve better results, all entries written in languages other than
English should be removed. Moreover, the proposed solution aims to benefit the software development community overall and since English is the most international language and the de facto standard within the software field, it stands to reason that the questions and particularly the answers provided by the system, are written in English. Besides, during the implementation phase, it became apparent that this filtering also had the added benefit of removing entries containing spam or just plain nonsense thus further improving the quality of the dataset which in turn increases the probability for the machine learning algorithms to produce accurate results. In order to determine whether an issue report is in English, both the title and the body were examined for containing common English words - if neither of the two have any, then the post is most likely not in English or it is spam or just a very poorly formulated inquiry - in either case, the post does not provide much value and is therefore discarded. The list of words utilized for this purpose is often referred to as a “stopword” list and is usually used for filtering words that do not carry much information about the actual meaning of a text. However, due to their ubiquity they are an excellent indication of whether a piece of text is written in English or not and thus perfect for the task at hand.

In the end, as a result of all the data cleaning activities, the issue dataset was further reduced to 30577 issues in total.

5.1.3 Data preprocessing

Even though the dataset had been thoroughly cleaned, there were still a number of preparation steps that could be undertaken so as to improve the quality of the data. These measures could help remove any “noise” from the data i.e. parts of it which are not particularly important, since they could have a negative effect on the issue categorization. For this purpose, a dedicated NLP library called nltk\footnote{http://www.nltk.org/} created by Bird, Klein and Loper (2009) was used, which is an excellent tool that provides a multitude of features for working with text.

Since, as mentioned in Section 4.1.1 the issue analysis is solely focused on the title, the first step was to extract that particular field from the dataset. Then, all characters of the text were converted into lowercase, which ensures that the same words or letters are not perceived as being different because of their capitalization. Next, any possible “noise” was removed from the titles, such as HTML tags, URL’s, duplicate characters and so on. Since the issues addressed different types of programming problems, such data is quite a common, but could serve to bias the clustering results, by giving unwarranted importance to inconsequential terms.

One of the most powerful features of the nltk library is its “tokenization” facility which is used for breaking up text into separate tokens, by using some sort of a delimiter, which in
most cases is a white space. Nevertheless, the tokenization process does not consider only words but all other non-alphabetical characters such as punctuation, numbers and so on. However, all of these special characters do not carry any meaningful information that can be used for determining the similarity among issues, and were therefore removed.

After splitting the title into separate words, the dataset was submitted to a process known as stopword removal. “Stopwords” are common words in a given language, in this case English, which do not provide any information regarding the actual meaning of the text. Such words include personal pronouns like “I”, “me”, “we”, etc., articles like “the” and “a”, conjunctions and so on. The removal process is accomplished by examining each word in a piece of text and checking if it is present in a so-called “stopword list” which contains a number of these terms, in this case a list consisting of a total of 153 words\textsuperscript{16}. In fact, this very same list was also used during the data cleaning phase described in Section 5.1.2, in order to determine whether a given post was written in English or not, however in this instance, the list is employed according to its original purpose, namely filtering out words that do not help determine the meaning of a given text document.

Even though stopword removal is a common preprocessing approach in natural language processing and text analytics, as illustrated by the research of Andrews and Fox (2007), Huang (2008), Xuan et al. (2015), Zhao and Karypis (2008) and so on, some authors argue that for some purposes, removing stopwords does not provide sufficient benefit and could even be detrimental. For example, Manning, Raghavan and Schütze claim that information retrieval applications such as web search engines often do not employ a stopword list, since some search queries are “disproportionately affected” (2009, p.27). Furthermore, they state that the cost of not discarding these words is not particularly substantial in terms of efficiency and processing time (Manning, Raghavan and Schütze, 2009 p.27). However, the dataset that is being analyzed in this instance is much more homogeneous - it is only a collection of JavaScript issues, compared to the enormous amount of diverse documents that are indexed by search engines. Therefore removing the stopwords would be much more beneficial in this context since they do not describe the essence of the issues and so by getting rid of them, it makes it more likely that similar issues will be grouped together, thus improving both the accuracy and precision of the proposed solution which is far more important than computational efficiency and fast processing times, as is the case for a typical search engine. Antoniol et al. (2008) also did not resort to stopword removal in their study aimed at classifying issue reports into one of two categories - those requiring corrective maintenance i.e. bugs and those that call for refactoring or enhancement i.e. non-bugs. Since they had these two binary categories that are in many ways conceptually opposite, this approach was well-suited because as they state, the existence of a stopword such as “not” would make a vital difference to the classification of a given issue. The same cannot be said for the prototype that is being developed as part of this research, again due

\textsuperscript{16}http://www.nltk.org/book/ch02.html#stopwords_index_term
to the fact that the presence of a relatively large number of categories necessitates a much more fine-grained distinction among the various entries, which can be better achieved by eliminating words that carry little or no contextual meaning.

The next stage of the process was to transform different variations of the same word into a common form - a prominent strategy in document similarity and text classification since it ensures that different word forms are treated as being the same, thus increasing the likelihood that documents containing similar words are seen as being related to one another. There are two techniques that are used to achieve this, namely stemming and lemmatization. Stemming is the process of converting words into their base form, also known as “stem”, which is often simply done by removing the suffix. Therefore, the result of stemming does not necessarily have to be a proper word as long as various derivatives of the same word all correspond to the same stem. On the other hand, lemmatization is a linguistic process that is used to transform words into their root form (“lemma”) through the use of a vocabulary and morphological analysis, which takes into account both the context and the meaning of the word. The context is provided by designating what part of speech a given word is - a noun, verb, adjective, etc., which can be achieved through a process known as part-of-speech (POS) tagging. Table 5.1 shows a comparison that illustrates the difference between the output produced by stemming and lemmatization.

**Table 5.1: A comparison of the different word transformations that occur when various verb forms go through the process of either stemming or lemmatization**

<table>
<thead>
<tr>
<th>Verb form</th>
<th>Stemming</th>
<th>Lemmatization</th>
</tr>
</thead>
<tbody>
<tr>
<td>study</td>
<td>studi</td>
<td>study</td>
</tr>
<tr>
<td>studying</td>
<td>studi</td>
<td>study</td>
</tr>
<tr>
<td>studied</td>
<td>studi</td>
<td>study</td>
</tr>
<tr>
<td>going</td>
<td>go</td>
<td>go</td>
</tr>
<tr>
<td>gone</td>
<td>gone</td>
<td>go</td>
</tr>
<tr>
<td>went</td>
<td>went</td>
<td>go</td>
</tr>
</tbody>
</table>

It is important to note that if the words were not tagged as verbs beforehand, all of them would have stayed the same after the lemmatization process, because due to the fact that no contextual information was provided, the lemmatization algorithm would incorrectly assume that these words are all nouns. So since each word is checked in a dictionary and none of them exist as nouns, the dictionary search would return no results and the output will be exactly the same as the input. Due to the fact that lemmatization requires all of this additional preparation in order to perform its function, it is therefore more time-consuming and computationally expensive, which becomes an even more important concern when dealing with large datasets. This could also be one of the primary reasons why stemming appears to be the more widely adopted approach across the field, as evidenced by the work of Andrews
and Fox (2007), Huang (2008), Zhao and Karypis (2004), Zhou et al. (2016), among others. Nevertheless, according to Han et al. (2012), despite the fact that stemming is the more efficient approach of the two, if precision is key to an application, as in the case of document categorization which is essentially the end goal of this whole process, lemmatization would be the more appropriate choice. Furthermore, due to the fact that lemmatization relies on morphological analysis, it is much better suited for establishing semantic relation between documents, as illustrated by Kashyap et al. (2016) who incorporate lemmatization as part of their extensive semantic textual similarity system which focuses on determining similarity between short text passages, such as the ones being used here (i.e. the titles of the issues). Last but not least, since the actual words (or features), that are extracted from the dataset would be needed during the expert analysis phase in order to help the examiners assign labels to the various document clusters, stemming is the less desirable solution because sometimes after being stemmed, some words become so obfuscated that it is challenging to figure out which word does the resulting stem actually represent.

Finally, if after undergoing this whole process, the content of an entry is entirely discarded (resulting in an empty string), which can occur if the title contains only common stopwords and special characters, it is removed from the dataset. As a result of all these preprocessing steps, the dataset was yet again reduced to a total of 30255 issues.

5.1.4 Feature extraction

As pointed out earlier, in order to identify distinct categories, the collected issues are separated into several groups consisting of related items through clustering. However, clustering algorithms cannot work with textual data and therefore, all text documents have to be transformed into a numerical representation. This is achieved through a process known as “feature extraction” which aims to identify the most relevant terms within each document in the dataset and present every document as a vector matrix which has the same size as the number of features that have been extracted. As mentioned in Section 4.1.3 this method is known as the vector space model (VSM). The transformation of issues into a vector representation was done through the use of a library called scikit-learn, which is a machine learning library developed by Pedregosa et al. (2011) that offers a wide range of algorithms for many different learning scenarios, such as classification, clustering and so on. Due to its relatively low learning curve as well as the excellent functionality that it provides, the library is being utilized by a number of big and influential companies such as Spotify, Booking.com, OkCupid, etc.

However, in order to determine the importance of various terms within a document dataset, the vector space model has to be further complemented by a term weighting function.
which tries to assess the significance of each word. For this purpose, the so-called “TF-IDF” function was utilized, which stands for term frequency - inverse document frequency. As the first part of the name suggests, the function takes into account the frequency of a given term in a single document. However, since documents differ in length, some terms may appear more often in longer texts than in shorter ones. Therefore, the term frequency is not merely a count of the total amount of times a given word appears in a text, but is instead the result of the number of occurrences of a word divided by the total number of words in the text. This ensures that the TF values are comparable among documents with varying length. Nevertheless, the problem with only using term frequency to determine word importance is that common words in the English language are going to appear in nearly every document despite the fact that they do not actually describe the content of the text. This is where the inverse document frequency comes in. In order to address this issue, the inverse frequency gives a higher score, and thus more importance, to terms that appear often in a small subset of documents within the dataset but rarely throughout the entire collection as a whole. This means that the weighting function assigns a score to each term which denotes its relevance, with a higher score indicating a more important term. This score becomes particularly crucial in determining document similarity because it helps establish how closely related two documents are. For example, if three documents share two terms each, as Table 5.2 illustrates, the two entries which share a more important term will be regarded as being more closely related. So in this case, since document 1 and 2 share the term “jquery” which has the highest TF-IDF score, they are considered to be closer to one another, than document 2 and 3 or 1 and 3, which also share two terms apiece, “function” and “array” for the first pair, and “size” and “function” for the second respectively. However, both of them have a lower TF-IDF value so they have a lesser impact on determining similarity.

Table 5.2: An example illustrating the TF-IDF weight scores which may be given to some of the terms shared among several text documents

<table>
<thead>
<tr>
<th></th>
<th>jquery</th>
<th>size</th>
<th>function</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document 1</td>
<td>0.726</td>
<td>0.69</td>
<td>0.13</td>
<td>0.0</td>
</tr>
<tr>
<td>Document 2</td>
<td>0.726</td>
<td>0.0</td>
<td>0.13</td>
<td>0.54</td>
</tr>
<tr>
<td>Document 3</td>
<td>0.0</td>
<td>0.69</td>
<td>0.13</td>
<td>0.54</td>
</tr>
</tbody>
</table>

The TF-IDF weighting scheme has been demonstrated to effectively identify the most relevant words within a given document and has become a de facto standard in the field of text document analysis, as shown by the works of De Boom (2015), Huang (2008), Zhao and Karypis (2004) and so on.

During the feature extraction stage, it is also possible to specify the maximum number of features that are to be extracted as a result. The benefit of this approach is that only the most relevant features are being considered. In the case of scikit-learn’s implementa-
5. Implementation

The relevance of a given term is determined based on its frequency across the dataset. Furthermore, selecting only a small, but descriptive subset helps eliminate any “noise features” (Manning, Raghavan and Schütze, 2009 p.271) which would serve to deteriorate the accuracy of the clustering. However, if the text documents being processed are small in size, as is the case with the issue titles, specifying a low feature threshold would result in a large portion of the documents being represented as vectors of null values, since they do not contain any of the extracted features. As a result, such documents would be considered as being identical during the subsequent clustering process and would be bundled together even if they are not really related to each other. On the other hand, one can instead let the algorithm choose how many features to extract. This minimizes the possibility that a document will be represented as a matrix of null values. The downside of this approach is that some words that do not carry significant meaning could be selected as features, although this can be remedied either by employing a stricter, more comprehensive stopword list or defining a lower and upper occurrence limit, which is discussed in the next section. Nevertheless, despite this trade-off, the latter option of allowing the algorithm to extract as many features as it sees fit was utilized. For the purposes of this study, if any null documents still exist, they will be discarded because the fact that these elements are presented as null vectors, even though there was no feature threshold limit, means that none of the words contained in them were considered relevant enough to qualify as features which in turn most likely signifies that the entries themselves were either of poor quality or addressed a very specific type of issue.

Another option that can shape the output of the feature extraction is defining what the algorithm should regard as features - single words i.e. unigrams, two word phrases (bigrams), three word phrases (trigrams), etc. or a combination of them. Even though as Lin and Wu (2009) argue, using whole phrases instead of singular words helps provide additional context and thus alleviate ambiguity, it is not a suitable alternative when working with short text passages because due to the limited size of the target content, entire phrases occur rarely and thus produce a small number of features, which leads to the aforementioned problem of having a large portion of documents being represented as vectors of null values. On the other hand, using a combination of unigrams and multigrams can result in a bigger number of features since two words can appear both separately and together as part of a phrase. What is more, documents sharing entire phrases are even more likely to be clustered together because they will have even more features in common. For instance, if two documents contain the same two word phrase, if multigrams are included in the feature extraction, then there are three features in common between the documents - the two terms on their own and as a pair. That is why the decision was made to use unigrams, bigrams and trigrams to form the features that are to be extracted.

As mentioned in Chapter 3, the overall research process had an iterative nature, meaning that as new information was being discovered, it was possible to go back and make
adjustments to different parts of the study in order to accommodate the newly found data. One example of this was the feature extraction phase which was further enhanced based on the results of subsequent stages in the workflow, namely the document clustering step which indicated that there were several words which were rather generic and did not provide much information regarding the core of the issue, but since they appeared in a number of entries, had a significant influence on the resulting clusters. Since these words were not very meaningful and only served to bias the clustering, the best course of action was to discard them prior to the training of the clustering model. As a result, the issues were separated based on other more relevant features that better describe the nature of each inquiry.

Just like the nltk library, the scikit-learn vectorizer, which is its dedicated facility for transforming text documents into numerical vectors, also allows for removing stopwords from the dataset and even offers its own stopword list consisting of around 320 words. However, one can choose to specify a custom list of words to be removed instead of using the default, which has the benefit of discarding words that may not be typical English stopwords but rather domain-specific stopwords - words which do not provide much information that could be used to distinguish a given entry from the rest. Since the issue dataset is quite homogeneous given the fact that all entries concern JavaScript related programming inquiries, there are a number of words which could be classified as domain-specific stopwords and thus removed from the data so that they are not considered during the feature extraction. Nevertheless, it is important to not be overly strict, due to the relatively small size of the processed issue titles. Therefore, the custom stopword list only consisted of very general terms which were found to be commonly used in user-submitted questions during the analysis of the initial clustering results, namely “problem”, “issue”, “help”, “work” and “bug”. The first three words are often utilized when a user has encountered some sort of an obstacle and is looking for a solution. However, the words themselves do not reveal anything about the nature of the problem and since they are bound to appear often in these types of inquiries, they can influence the cluster assignment of the entries which contain them. If the dataset was more diverse and covered a wide range of topics such as programming, politics, religion, sports, etc., these terms would have been extremely valuable for determining the category of an entry, but since, as mentioned, the entries are pretty similar, such words could have an adverse effect instead. The term “work” is also closely related to these words, because it is mostly used to convey the fact that something does not work or it works in an unexpected fashion. However, again, the word does not capture the essence of the problem and is therefore discarded as well. Finally, the term “bug” is also removed because determining whether something can be regarded as a bug or not is rather subjective. For example, a word such as “error” which could also be seen as a generic term is used much more accurately because it is often contained within a feedback message returned by the system, meaning that one can confidently say he is getting an error. On the other hand, a bug does not necessarily manifest itself in such an obvious way and so a
lot of the time it is difficult to pinpoint its existence. Last but not least, as Herzig, Just and Zeller (2013) discovered, bugs are very often misclassified as such by users, meaning that the presence of the term “bug” in an entry could actually be misleading. Therefore, all of the aforementioned 5 terms were removed from the dataset before conducting the feature extraction. This sort of incremental improvement to different parts of the solution is one of the chief benefits of employing an iterative methodological approach and can ultimately help create a better, more complete product.

As noted earlier, in order to further filter the dataset and ensure that the resulting features are in fact the most relevant and meaningful terms describing the core of the underlying issues, it is possible to ignore words that appear too often or too rarely. For example, if a term appears in many documents, then it is either some sort of a stopword that was not detected or another domain-specific word that does provide some information about the content of the text but it is still much too generic for it to be useful. Conversely, if a term appears too rarely across the entire dataset, then it is not of significant importance. That is why, if a term appears in more than 5% of the documents in the dataset or fewer than 10 times in total i.e. it is present in less than 0.03% of entries, it is not considered as a feature. The 5% upper threshold might seem like an extremely low bar, but since the dataset is rather large, 5% would represent over 1500 issues. Furthermore, the input text is quite short so a word that appears more than 1500 times, without being particularly descriptive, would only serve to bias the clustering. It is important to mention that this number was not randomly selected but instead settled upon after some experimentation which uncovered that setting a higher threshold did not eliminate any words whatsoever, so the bar kept lowering until reaching the 5% level. The 5% threshold served to eliminate two very generic terms which were likely to appear in any type of issue. Lowering the bar below that point, resulted in the algorithm starting to discard terms which could be considered relevant and so the limit was set at 5%. Table 5.3 shows the two words that exceeded this threshold, both of which do not provide any specific information regarding the nature of the issue. It should be noted that these words are only discarded as single entities (unigrams) but they can still exist within the feature set as part of a multi-word expression (bigrams or trigrams). For example, despite the fact that the word “error” is not considered as a feature, the expression “memory error” might still be regarded as such.

The lower threshold was also settled upon after some experimentation, which showed that removing words and expressions that appear less than 10 times in total, takes care of the large majority of inconsequential terms. Moreover, setting the bar higher, resulted in discarding a number of words which would be quite relevant in describing the nature of the user inquiry. Table 5.3 presents a small sample of the terms that were omitted, not including all the bigrams and trigrams that were ignored even though they comprised the majority of discarded items. These terms include misspellings (which made up a large portion of the data), words in foreign languages that somehow managed to slip through,
abbreviations, names of specific functions or technologies and so on. Naturally, there are also regular words, which do indeed carry meaning regarding the posts in which they appear, but since they did not meet the predefined minimum occurrence threshold, they are also being removed. Of course, the exact lower limit is not set in stone and could be tweaked if needed, depending on one’s purposes.

Table 5.3: A sample of the terms excluded from the feature dataset because they appeared too often or too rarely

<table>
<thead>
<tr>
<th>Terms in over 5% of documents</th>
<th>Terms found under 10 times in total</th>
</tr>
</thead>
<tbody>
<tr>
<td>error, use</td>
<td>atleast, 自定义工具提示在特定情况下失效, maven, und, interrupt, purpose, adb, chllenge, getimage, параметров, unhandledpromiserejectionwarning, data-zoom缩放的中间过程中, adapter, instead, actual, unlike, oldval, yxz, fetchstart, td, printing, completly, twilio, adjust, enhancedswitch, scalability, gittleparser, boowho, gulpfiles, reporter</td>
</tr>
</tbody>
</table>

The feature extraction process resulted in 2170 features containing unigrams, bigrams and trigrams that have appeared at least 10 times throughout the dataset and in less than 5% of the total documents. Therefore, each of the 30255 issues was now represented as a vector of 2170 dimensions, with each value corresponding to one of the extracted features. So if a given document contains a feature, the value at its index shows the TF-IDF score of the feature, while if the feature is not present in the document, the value at that index is null. After the feature extraction stage, there were still a total of 928 null documents, i.e. documents containing solely null values since none of the extracted features appear in them. As mentioned, due to the fact that despite the rather large number of features that were extracted, these issues still did not contain any of them, this indicates that they are either poorly written by the reporter or they pertain to a very narrow topic. In either case, because of their vector representation, they will inevitably fall into the same category, thus being of little value. Therefore, they were all removed from the dataset. Of course, this has the inherent limitation that some genuine, well-written entries regarding a specific type of issue will not be categorized. However, this problem can be addressed in future iterations of the implementation, if the chosen approach shows to be promising.

After this operation, the total amount of issue vectors ready for clustering was 29327, meaning that over 92% of the original data that was gathered was still present, indicating that it was of relatively good quality to begin with. Since the feature extraction process is deterministic, given the same parameters, the output will be the same as well, meaning that the entire procedure can be repeated several times if needed, without affecting the end result.
5.1.5 Feature clustering

After the dataset had been thoroughly cleaned and prepared and the desired features had been extracted, the next step was to run a machine learning algorithm using the selected features in order to train a model. Again, as mentioned on several occasions, the exact configuration of the solution is not the focus of the research and therefore questioning the suitability of the chosen clustering algorithm is not a primary concern. Having in mind the task at hand and the specific goals that the research aims to achieve, the K-Means algorithm was determined to be a good fit for the purpose and was thus incorporated as part of the prototype.

It should be noted that apart from the regular K-Means implementation, scikit-learn also offers an alternative called a Mini-batch K-Means, which is a version of the K-Means algorithm that is more suited for the demands of the web. The Mini-batch K-Means was proposed by Sculley (2010) in order to address some of the challenges posed by modern web and mobile applications. The way it works is the algorithm takes a random sample of the input data (a mini-batch) of a predetermined size and assigns all the data points within the batch to the most suitable cluster. Then, with each iteration, it takes a new randomly selected batch of the same size and uses it to update the clusters. This process is repeated until the algorithm converges or the specified number of iterations is reached. This approach serves to significantly reduce both the computation time and cost since the algorithm does not need to load or process the whole dataset, which makes it an ideal solution for working with large amounts of data. Moreover, as Figure 5.1 illustrates, the results of the clustering are virtually identical to the ones produced by the traditional K-Means algorithm.

![Figure 5.1](image.png)

**Figure 5.1:** Comparison of the difference in the clustering results between the K-Means and Mini-batch K-Means algorithms, when applied to the same dataset (Source: scikit-learn, n.d.)

The Mini-Batch K-Means is primarily intended for scenarios where the clustering is performed “on the fly” and so the time needed for the process to complete is of utmost importance. However, the proposed solution is designed in such a way that the clustering
only needs to be performed once and from then on, the trained model can be used for further manipulations. Therefore, in this context, the main advantage of the Mini-Batch K-Means loses its significance compared to the regular K-Means algorithm and so the latter was chosen to be employed for the clustering instead.

Since each element in the dataset is represented as a numerical vector containing as many values as there are features, text documents generally have a high number of dimensions which introduces a number of challenges. This phenomenon is commonly referred to as “the curse of dimensionality” and according to Tang et al. (2005), is “the most critical problem for document clustering”. One of the most significant obstacles introduced by the large number of dimensions is that, in high dimensional space, the distance functions used to determine object similarity or dissimilarity are no longer a meaningful criterion because due to the large dimensionality of the data, the distances between points become virtually “indistinguishable” (Assent, 2012). What is more, since any given document contains only a handful of the extracted features, the data itself is sparse, meaning that the large majority of dimensions contain null values and are therefore irrelevant in terms of describing the document. Hence, they would not be useful in any subsequent analytical tasks, and actually can have a negative effect on such activities (Houle et al., 2010). In fact, Houle et al. (2010) argue that the presence of irrelevant features has a more detrimental influence on the clustering results of high dimensional data than the ineffectual distance measures.

Different techniques have been applied in order to address the aforementioned problems, with the most prominent being what is called “dimensionality reduction”. The goal of this approach is to “find a lower dimensional representation of the data while minimizing information loss” (Assent, 2012). Essentially, what this technique does is it tries to identify the most prominent features that best describe a given item and uses them to reduce the dimensionality of the entry without altering its core structure. Moreover, as stated by Dumais (2004), by decreasing the total number of data dimensions, this method helps deal with two prominent challenges in natural language processing, namely synonymy and polysemy. Synonymy refers to the trait of different words to have the same meaning, such as “couch” and “sofa”. On the other hand, polysemy is the characteristic of words to have multiple meanings, for instance the word “plane” or “interest”. However, lowering the dimensionality of the data helps alleviate the effects of these problems by making it easier to uncover similarities among different terms (Dumais, 2004). Furthermore, since there is less data to be processed, the subsequent analysis tasks naturally take a shorter amount of time to complete.

As Assent states, the amount of information that is lost during the reduction process largely depends on the “inherent dimensionality of the data (i.e., the number of dimensions that represent noise) and the degree of reduction” (2012). According to Manning, Raghavan and Schütze, dimensionality reduction can be viewed as “soft clustering” (2009, p.417), where each dimension of the reduced vector space represents a separate cluster and the
document value at that dimension signifies its implicit membership to a given cluster. One example of a specific dimensionality reduction method is latent semantic analysis (LSA), also referred to as latent semantic indexing (LSI) in the context of information retrieval. Latent semantic analysis is a statistical approach that attempts to infer the underlying meaning of documents and their relationship, based on the words within them, as well as the meaning and relationship of the words themselves, based on their co-occurrence in different documents and contexts, without relying on any morphological analysis or dictionary reference (Dumais, 2004). LSA functions similarly to the vector space model described in Section 4.1.3 and can be directly applied to a vector matrix consisting of term weights, such as the one generated as a result of the feature extraction process explained in Section 5.1.4.

A study conducted by Tang et al. (2005) suggests that LSA produces best results at lower dimensionalities. In fact, the researchers state that this technique can “effectively reduce the dimensionality” of a dataset from several thousand dimensions to a range of 100-200 or less (Tang et al., 2005). This is corroborated by the research of Schütze and Silverstein (1997), who discovered that not only is there no loss of quality in document clustering results when utilizing LSA for dimensionality reduction, but also that there is no significant difference between using 20, 50 or 150 dimensions. The authors argue that this phenomenon is caused by the fact that clustering is a “less fine-grained task” (Schütze and Silverstein, 1997) that does not require an extremely high level of precision, which allows for the dimension reduction to be more severe. Therefore, based on previous research efforts, the feature dataset is to be reduced to 100 dimensions using the LSA technique. If the resulting clusters are not of satisfying quality, the reduction process along with the clustering itself, will be repeated again with different parameters.

Another important concern in document clustering, and clustering in general, is the measure used for determining whether different elements are similar or not, so they could be correctly assigned to an appropriate cluster. By default, the K-Means algorithm uses the Euclidean distance metric (Huang, 2008). However, there appears to be a strong consensus within the field that cosine similarity is a more accurate measure for text document clustering tasks, because it can better recognize the relation among different documents. As a result, it has been utilized in a number of studies, such as the ones conducted by Lin and Wu (2009), De Boom et al. (2015) and Kashyap et al. (2016). Furthermore, Huang (2008) conducted an empirical investigation comparing several similarity measures used for document clustering across seven text document datasets and found that cosine similarity consistently produces more accurate clustering than Euclidean distance. One of the main advantages of the metric is that, unlike Euclidean distance, it also takes into account the length of the documents when performing comparisons between them. Moreover, it can efficiently work with sparse data, thus yielding fast computation times even when processing large datasets (Dhillon and Modha, 2001). Due to the various benefits that cosine similarity provides, Dhillon and Modha proposed a version of the K-Means algorithm which makes use
of this measure, named “spherical K-Means” (2001), which has since been often adopted in other studies, such as the work of Dhillon, Guan and Kogan (2002) as well as Buchta et al. (2012), to name but a few. However, since the K-Means algorithm uses distance measures in order to assign data points to a given cluster, a small distance value between two points signifies that they are related to each other. On the other hand, the opposite is true for similarity measures, where high values indicate close relation. Therefore, in order to utilize cosine similarity with the K-Means algorithm, the similarity measure has to be inverted, thus resulting in a cosine distance (or cosine dissimilarity).

Unfortunately, despite the fact that it is a very feature rich library, scikit-learn does not provide a clustering implementation that utilizes cosine distance. However, the natural language processing library that was used for preparing the document dataset for the feature extraction and subsequent clustering, namely nltk, does have such an implementation. Nevertheless, after some testing, it became apparent that the aforementioned solution was taking an excessive amount of time to complete which proved to be a considerable obstacle during the development process. Upon further inspection, it appeared that the cause of this issue was the implementation itself, since it was not realized very efficiently. Most clustering solutions allow one to define the maximum number of iterations that the algorithm should perform even if it has not yet converged i.e. it has not reached a stage where changing the cluster centers yields no change in cluster assignments, because that might take a very long time. This problem can be addressed by setting a more generous convergence threshold defining the maximum level of change that can be neglected, however, this would have a detrimental effect on the quality of the clustering. Furthermore, since scikit-learn is a library specifically designed for handling machine learning tasks, it has a much more advanced and sophisticated implementation than the one available in nltk, whose main purpose is, of course, natural language processing. One example of this is the so-called “K-Means++” initialization method which can be used in scikit-learn for selecting the initial cluster centers. K-Means++ is an improved version of the K-Means algorithm proposed by Arthur and Vassilvitskii (2007) which aims to avoid some of the inherent shortcomings of the K-Means algorithm and thus optimize the clustering results. Essentially, what the algorithm does is enhance the “seeding” process i.e. choosing the initial data points which serve as cluster centers. Instead of selecting them completely at random, as does the traditional K-Means algorithm, K-Means++ chooses only the first cluster center in such a fashion and then selects all other centroids using a distance calculation which ensures that the resulting centers are spread out. According to the authors, despite the fact that the seeding procedure takes slightly longer to compute, this approach serves to improve both the speed and accuracy of the clustering, “often quite dramatically” (Arthur and Vassilvitskii, 2007). Last but not least, since the TF-IDF vectorizer of scikit-learn automatically normalizes the documents in the dataset, all vectors are of the same length, which effectively eliminates the aforementioned chief advantage of the cosine similarity measure compared to the Euclidean
distance, thus making the two metrics equivalent to one another (Schütze and Silverstein, 1997).

As mentioned in Section 4.1.2, the number of clusters which was determined to be suitable for the training was 30, and so the K-Means algorithm ended up dividing the vector dataset into 30 separate categories signified by a cluster index, which does not carry any real meaning but merely serves as a form of unique identification for each cluster.

5.2 Issue assignment

After the clustering model has been trained, the next step is to ensure that it can be utilized for assigning new issues to one of the formed categories. Not only will this provide another way to assess the accuracy of the clustering model but it is also an essential prerequisite for finding potential solutions to a user query, which will almost certainly be new and unique, from the perspective of the system. Even though, a trained clustering model created in scikit-learn, provides an issue assignment functionality by default, there are a number of preparation steps which have to be completed before it can be utilized. Therefore, this section provides an overview of this procedure and explains each step in more detail.

5.2.1 Data preparation

Essentially, any newly submitted issue has to undergo the same process as the entire dataset that was used for training the clustering model. First, it goes through a thorough preprocessing including tokenization, stopword removal, POS tagging and lemmatization, as described in Section 5.1.3. Then, the title is transformed into a vector representation, again using the TF-IDF weighting scheme. However, since the new entry only contains a few words, simply transforming it into a vector would result in a representation that has a much smaller number of dimensions compared to the original issue dataset. Furthermore, the TF-IDF score for each term will be evaluated on the basis of the current input and therefore, terms will receive completely different TF-IDF values than the ones they had during the training process. As a result, it would not be possible to accurately compare the new issue title with the data that was used for the clustering, which would have a detrimental effect on its classification. Nevertheless, there is a rather simple solution to this problem. During the feature extraction stage described in Section 5.1.4, the resulting 2170 features that were discovered and used for representing each document in the collection, were saved as a “vocabulary” which can be utilized for transforming other textual data into a vector that has the same length as the vocabulary itself. This in turn makes the new entry comparable to the documents used for building said vocabulary. Therefore, using this vocabulary, the new document can be converted into the vector space as if it was part of the original data. This means that all terms within the new entry which are part of the extracted features, receive the same TF-IDF scores and thus can be used for determining
in which cluster the new issue should be assigned. Of course, since the dimensions of the original dataset were reduced before performing the clustering, the new item also has to undergo the same procedure before it can be assigned to a particular cluster.

5.2.2 Classification

Once the new issue is transformed into a vector representation and reduced to the same number of dimensions that were used for training the clustering model, it is ready to be classified to a given cluster. The assignment itself is done by comparing the new vector with the centers of all the available clusters and determining which of these points is positioned closest to the new entry in the vector space. It should be noted that the clustering model does not have to process new vectors one by one, but can also do it in bulk - performing the aforementioned check for each vector and assigning it the index of the corresponding cluster.

One way to test the logic of the issue classification process and ensure that everything is operating correctly, is to select a random sample of issues from the dataset used for training the clustering model, put them through the same procedure that a newly submitted issue would go through and check whether they are assigned to the same clusters as before. Since the entire process is the same, the input vector should be identical to the one used during the clustering and so the resulting assignment should also be the same. If that is not the case, this would mean that either some parts of the logic are incorrect or that some of the system components are not operating as expected.

This rather simple validation step turned out to be enormously beneficial in identifying weaknesses in the overall implementation and thus ultimately developing a better, more complete solution. This check not only helped discover a logical error in the vectorization process prior to the issue assignment (which was easily fixed), but also highlighted the fact that the dimensionality reduction facility that was utilized, served to introduce a drawback which made it unsuitable for the task at hand. In order to reduce the number of dimensions of a vector dataset that contains sparse input, which is virtually always the case when dealing with textual data, scikit-learn offers a module called TruncatedSVD\(^\text{19}\), which stands for singular value decomposition. However, by default the algorithm on which the module relies, is based on the implementation proposed by Halko, Martinsson, and Tropp (2011). This means that the algorithm uses a randomized component that aims to reduce the computation time required to decrease the size of the input matrix. This is very beneficial when working with large datasets, but due to the presence of the random element, each time, the output of the algorithm is slightly different despite the fact that it might be using the exact same input. This can be addressed by specifying a predetermined value called a “seed” which makes the randomization repeatable, meaning that given the same

seed, the resulting output will also be the same. Nevertheless, tests demonstrated that even when using an initial seed, the values of some vectors differed from the ones these same vectors had during the model training and so, as a result, they received a different cluster assignment. In some scenarios, where absolute accuracy might not be of paramount importance, the increased speed that the randomized SVD algorithm provides could justify its use despite the slight loss of precision. Furthermore, even in this particular context, this drawback could potentially be disregarded due to the relatively small margin of error (tests showed it to be around 1-2%, sometimes even less). However, since the reduced vector representations are only calculated once and the dataset itself, despite being of substantial size is not overly big, the speed considerations are not as crucial. Last but not least, the category assigned to a given issue has a significant impact on the solution suggestions presented to the end user and therefore it is of paramount importance that the assignment is both reliable and accurate. That is why, the prototype implementation was slightly changed in order to address this problem and ensure that, given the same input, the vectorization and dimensionality reduction facilities produced identical results on a consistent basis.

5.3 Solution suggestions

Even though the ability of the prototype to automatically classify an issue to an appropriate category is in itself a substantial contribution which brings a number of significant benefits, the aim of this research was to go one step further and demonstrate how this capability can be utilized to aid developers in their search for solutions to various programming issues. Two separate approaches have been implemented, both of which are described in the following sections.

5.3.1 Category-based suggestions

The first approach for providing relevant solutions consisted of several steps. First of all, the newly submitted issue is transformed into a vector and assigned to a given cluster, as described in Section 5.2. Then, the issues which are most similar to the query are found, by comparing the distance between the new vector and all other vectors inside the cluster. Finally, the comments posted for each of these similar issues, are extracted and the top rated ones are presented to the end user. As mentioned, the comments associated with each issue were gathered as part of the data collection stage of the research process, described in Section 5.1.1. The goal of this approach is to ensure that the search results are both relevant to the original query and useful in their own right. However, the problem with this solution is that some clusters are quite large and therefore finding the most similar issues within them and then processing all the comments related to each issue would require quite some time, especially if the dataset grows in size, which is of course the intention of the entire system - being able to work with enormous amounts of information. Therefore, in
order to address this concern, the clusters created during the model training, were further divided into multiple levels of subclusters, again using the K-Means algorithm, thus forming a sort of a hierarchical tree.

The hierarchy was created by first dividing each of the 30 clusters into another level of 5 subclusters and then if any of the resulting subclusters contained more than 100 issues within them, they were split into a third level, again consisting of 5 more subclusters. Since the lower level subdivision depended on the number of issues inside a given cluster, not all subclusters were broken down any further and as a result, the hierarchy tree had an asymmetrical shape, with some “branches” being more extensive than others. Naturally, top level clusters which had a large number of issues within them, were more likely to be divided into more subclusters than the ones which contained a smaller amount. What is more, the subclustering process revealed an interesting pattern, namely the fact that at the lowest level, issues were divided quite disproportionately inside the resulting clusters. In most cases, one of the clusters contained the majority of entries, while the rest had a rather small size in comparison. One possible explanation for this phenomenon is that the issues within the large subcluster share some sort of a defining characteristic that differentiates them from other types of issues. Therefore, this characteristic could potentially be used to describe the entire parent cluster and, by extension, the issues assigned to it. However, this hypothesis requires further investigation and could be addressed as a part of future research efforts.

The relatively small subcluster number that was chosen (i.e. 5), was intended to compensate for the rather large amount of top level clusters and thus create a more balanced hierarchy were none of the sublevel clusters end up being much too little (or too large). Furthermore, such a structure can help minimize the number of distance comparisons that have to be made so as to assign an issue to the appropriate cluster, while at the same time providing sufficient differentiation within the groups. It is important to point out that unlike the top level clusters, the subclusters were not necessarily meant to be meaningfully distinct compared to the others, but rather to serve as a means for more specific categorization of issues. In fact, this hierarchical structure of subcategories can be very beneficial for finding solutions to a user query, because when an issue is classified, it first receives the index of the top level cluster and then traverses down the hierarchy until reaching the bottom. At each level along the way, the issue receives the index of one of the smaller subcategories and so the answers displayed to the user, end up being the ones found within the final subcategory. For instance, let us say that an issue is assigned to cluster number 16, which has 2411 issues inside. Then, it is placed into subcluster number 2 which has 578 issues, before finally being assigned to sub-subcluster 0 which has 93 issues. As a result, only these 93 issues are being considered when searching for the most similar ones and subsequently presenting the top rated comments to the user. Of course, if the subcluster to which an issue is assigned does not have any further sublevels, all issues within said cluster are analyzed during the
5. Implementation

So after a user-submitted query has been classified into one of the available clusters and any of the variable levels of subclusters, the next step is to find which issues within that cluster share the biggest resemblance to the new one. This ensures that the resulting output is as relevant to the user query as possible. Determining how similar two issues are, is accomplished by calculating the distance between their vector representations - the lower the distance, the more similar the two vectors i.e. issues are. Therefore, finding the most similar vectors to a given input would require that input to be compared to all vectors within the collection and extracting the ones which have the lowest distance value to it. This means that the smaller the collection, the lower the total number of comparisons that are needed to be performed, which in turn decreases the amount of time required to conduct the whole operation.

Once the most similar issues have been identified, the comments posted as responses to these inquiries are retrieved. After that, all missing data is removed and any comments that are not written in English are also discarded. As mentioned in Section 5.1.2, the main reason for this is the fact that English is the de facto standard language used in software development - almost all programming languages have an English syntax, the documentation of various project is written primarily in English and so on. Therefore, showing answers that are written in English would be helpful to as many people as possible. After the comment dataset has been cleaned, the next step is to determine whether a given comment has obtained a positive reception from the community.

This is done through the use of the so-called “reactions” which act as a form of a rating scheme that allows GitHub users to vote on a particular comment and thus express their opinion about it. There are six different reaction types, as seen in Figure 5.2, ranging from positive such as “+1”, “heart”, “hooray”, “laugh” to negative ones like “-1” and “confused”. Even though these ratings are quite subjective, they present a sort of quantitative measure that can help assess whether a given comment is of high quality or not. For instance, Figure 5.3 shows an actual example of how reactions are used by the GitHub community to indicate their appreciation for an entry\textsuperscript{20}, which they perceive to be helpful or useful in some way.

\textsuperscript{20} https://github.com/Automattic/mongoose/issues/4291#issuecomment-230312093
Figure 5.3: An example usage of the GitHub “reactions”, which allow users to rate the quality of a given comment

So in order to find the top rated comments for a specific collection of issues, the reactions for each comment in the filtered dataset are aggregated so as to compute a total “positive” score, which would then be used to sort the results. Each positive reaction that a comment has received adds to its overall score, while a negative one reduces it. This would help to balance out the ratings and decrease the score of comments which might be seen as debatable or ones that are not universally accepted. What is more, since the goal of the whole suggestion functionality is to provide end users with helpful information that is relevant to their search queries, the “laugh” reaction is also considered as a negative one and thus serves to downgrade the overall score. The reason for this is that, this reaction most likely indicates a humorous entry which could be appreciated by the community but does not actually add relevant information to the discussion. Furthermore, if a comment is of genuine quality, all other types of positive reactions would compensate for any erroneous downgrading caused by this particular decision.

Finally, after the aggregated positive score has been calculated for each comment, the top 5 most highly rated comments are presented to the user, along with their respective score. The reason for showing only 5 results, is that the prototype aims to determine whether the suggested answers are in fact relevant to the user query and so, if the 5 best rated answers do not fit that criteria, it would be unnecessary to present multiple pages of results. It should be noted that the 5 comment threshold illustrates the maximum amount of entries that are to be displayed. If there are less than 5 comments which have a positive score of at least 1, then the results returned to the user will also be fewer, because without
the score, there is no indication of the comment quality. Of course, this means that if there are no comments with a positive total, nothing will be shown.

Figure 5.4 shows how the web prototype looks in its initial state, before any query has been made. As the diagram illustrates, the application has a similar visual appearance to that of a traditional search engine, which serves to provide an intuitive sense of how the prototype can and should be utilized, even to users who are not familiar with it.

![GitHub Issue Analyzer](image)

**Figure 5.4:** A screenshot illustrating the visual appearance of the web application prototype in its default state (when the page loads)

On the other hand, Figure 5.5 shows the output that is presented to the user, after submitting a search query. As mentioned, the best rated comments appear at the top of the search results, along with their respective positive score and textual content.

![GitHub Issue Analyzer](image)

**Figure 5.5:** A screenshot of the output (i.e. “solution suggestions”) produced by the web prototype as a result of a given search query

Each comment is also accompanied by a URL directly pointing to the original entry as well as the entire issue thread, so that users can see the response in context, evaluate
whether the initial inquiry is related to their own and examine the rest of the comment section. This design pattern is very similar to that of Google’s “rich answers”, which are short snippets contained within a small box that appears at the top of the search results which provide a direct response to a user’s query. As illustrated in Figure 5.6, the rich answers come in many shapes and sizes, depending on the particular question that has been submitted, but their general goal is to answer the inquiry without the user having to even browse to any given website.

![Figure 5.6: Two examples of Google's rich answers shown in response to different search query types](image)

One of the main reasons for the growth of these snippets on Google, is the fact that the people at the company have realized that often times people are just looking for quick answers to their questions, instead of doing in-depth research on a given topic. Even though, searching for programming related issues is a rather different use case, the benefits of the rich answer approach are obvious, since whenever users ask a question, their end goal is to receive a straight answer, meaning that more often than not, they would gladly skip the intermediary stage of looking through the results and having to click to another website, in favor of getting a direct response, which is what the prototype is offering by directly showing the contents of the different issue comments.

As stated earlier, the creation of the cluster hierarchy was driven by the assumption that, due to the large number of issues within some of the clusters, finding the most similar ones and then identifying the best rated comments within them would take an excessive amount of time, particularly when the dataset that is being analyzed increases in size. Naturally, as you go down the hierarchy tree, there are less and less issues to be processed, and respectively
less comments that are being considered. Since the K-Means clustering algorithm groups vectors based on distance, chances are that the issues within the bottom subclusters will be more similar to the user query than the ones found at the upper cluster levels. Therefore, there is this juxtaposition between issues, and by extension, comments/solutions that might be more relevant but fewer in number and issues/solutions which might not be as pertinent, but present a larger pool. What is more, even entries which are not directly related to a given query, might still provide some value or aid users in their search for a solution. As a result, instead of deciding which type of suggestions should have a precedence, on behalf of the users, the web prototype incorporates a functionality which allows them to make that choice for themselves. By default, the prototype displays the best rated comments of the most similar issues found in the first sublevel of the clustering hierarchy but, as shown in Figure 5.7, it also includes a range slider which can be used to adjust this setting.

![Figure 5.7: A screenshot highlighting the suggestion range slider, which specifies how closely related to the search query should the output of the application be](image)

The slider has three possible values signifying the desired precision (or relevance) of the results, ranging from “high” to “moderate”, with each of these options not only representing the fact that the search is conducted at a different level in the hierarchical tree, but also specifying different criteria that is used for determining issue similarity and finding the top rated comments. For example, as shown in Figure 5.8, the “high” precision option considers the bottom i.e. third subcluster level of the hierarchy, which is marked in green on the diagram, and defines a rather low distance threshold that a vector (issue) should pass in order for it to be considered as similar to the search query. Furthermore, it also designates that the comments should only be extracted from the 15 most similar issues, which are under the aforementioned similarity threshold. Again, as is the case with the comments themselves, the specified limit of 15 indicates a maximum amount, meaning that if less issues are found to be similar enough, only they will be shown to the user.
5. Implementation

Figure 5.8: A diagram illustrating all the levels of the clustering tree (marked in green), which are being considered for searching potential solutions, depending on the currently selected “precision” setting.

On the other hand, the “moderate” precision option examines the top cluster level and specifies much more lenient search rules, by defining a higher distance threshold and allowing the comments to be gathered from the top 50 issues that pass it. Lastly, the default slider option, which is positioned in the middle of the range, traverses the first sublevel of the clustering hierarchy, specifies a more balanced similarity threshold and sets the issue limit at 30. This functionality allows users to not only search more effectively, but also offers them an additional degree of flexibility that can improve their chances of finding a relevant solution. For instance, if the high precision search does not yield any results, then changing the slider to a different value will widen the search range and potentially produce a useful response.

The main shortcoming of this whole approach is the fact that the ratings used for the aggregation do not represent real time up-to-date information but instead reflect the state that a given comment had at the time of the original data collection. Since then, the rating of the comment could have changed (it may have received more upvotes and/or downvotes), other comments could have been posted that have acquired a higher score and so on. Unfortunately, none of this is taken into account and therefore the results which are displayed to the end user might not accurately represent the current state of affairs. Nevertheless, since each result also includes a link to the original source, users have the option to examine the answers in more depth, which helps to at least partially lessen the effects of this problem, because by following the link, users will be able to see an up-to-date version of the entry as well as the whole discussion. However, if the comment has been removed or severely edited that might cause a new set of problems by being more confusing.
than helpful, which is something that also has to be taken into consideration.

5.3.2 Similarity-based suggestions

Apart from showing the most upvoted comments that were extracted, the application also presents the user with a list of issues which were found to be most similar to the submitted query, but were not included in the main search results. Of course, none of the similar issues are directly shown as a part of the suggestions, but since each comment that is displayed also includes a link to the original source, users can examine the entire issue thread and thus see what the initial inquiry was about and how it relates to their own. However, due to the fact that only the top rated comments are being presented, not all of the issues which were found to be similar are necessarily included in the output. Therefore, from this perspective it could be argued that the results are somewhat incomplete and additional measures could be taken to ensure that they provide sufficient value for the people using the application. What is more, showing a list of similar issues offers a number of additional benefits. First and foremost, it allows users to examine other entries which are presumably related to their own, which facilitates the process of finding a relevant solution. Furthermore, it can serve as a way for users to evaluate the quality of their own query - if the suggested issues contain similar terms but are conceptually different, it can indicate that the inquiry itself requires some restructuring. On the other hand, if the issues are relevant, it offers users a chance to find a helpful answer that was not included in the list of suggestions, as well as to navigate through the whole issue thread and view the entire discussion.

Moreover, the solution suggestion approach outlined in the previous section has one major shortcoming. If the clusters that were formed during the model training, are not that well-separated, a newly submitted vector might end up being closer in distance to vectors assigned to a different cluster – a phenomenon caused by the way a new vector is assigned to a specific category. As mentioned in Section 5.2.2, the vector is compared to the centroid point of each cluster and then given the same cluster index as that of the closest centroid. Therefore, if the coordinates of the new vector happen to be somewhere near the edge of the cluster, which, as stated above, is positioned in close proximity to another cluster, then the vector might be closer in distance and thus more similar to vectors from the neighboring group. Nevertheless, since the cluster-based similarity comparison approach only takes into account issues that have been assigned to the same cluster as the new vector, none of the other issues are even being considered. Hence, the results shown to the end user might not necessarily be the most relevant ones. Figure 5.9 presents a diagram\footnote{Adapted from: \url{http://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html}} which attempts to visualize this very problem.
The two data points inside the circled area, which represent two vectors, are closer in relation to one another than to their respective cluster centroids, which are signified by a black “X” mark. However, since the red data point is closer to the centroid of the red cluster than the centroid of the green one, it is assigned to the red group. The same applies to the green data point which is closer to the green cluster centroid than the red. Therefore, if a new data point appears, and its position happens to be somewhere close to the edge of its cluster, then the same situation could possibly occur. Nevertheless, as mentioned, this phenomenon would not take place unless the clusters are in close proximity to each other. For instance, if we examine the orange and the green clusters, each point in both categories is closer to its respective centroid than to any point within its neighboring cluster.

The most straightforward way to address this problem would be to take the clustering out of the equation and compare the new vector with all other vectors within the entire dataset, which would help identify the most similar ones, with absolute certainty. However, as mentioned, since the vector similarity is estimated by direct distance comparison, the total number of vectors determines the amount of comparisons that are to be made, which in turn dictates how long it will take. This means that decreasing the number of comparisons that have to be performed in order to obtain a result, as is the case with the clustering hierarchy method, would be a much better option if scalability is of paramount importance. Therefore, these two approaches - utilizing clustering to form smaller yet related groups
which can be processed more easily, and performing a distance comparison using the whole dataset, present an interesting trade-off between accuracy and performance and choosing which one of the two should be adopted may depend upon evaluating which attribute is more vital to the application.

However, after conducting some tests, it became apparent that the scikit-learn module used for distance comparisons is very efficient and can perform a large number of checks very quickly. In fact, it is able to compare a new vector to all 29327 vectors inside the dataset in less than 0.7 seconds, although it should be pointed out that the computation time would most likely become more significant as the number of comparisons increases. Therefore, this approach might not be able to scale well and could become problematic as the data that is being analyzed grows in volume. Nevertheless, due to the efficiency of the distance comparison implementation, the prototype is not affected by the aforementioned trade-off, meaning that the application can benefit from the increased accuracy that this approach provides, without suffering any of the negative consequences, at least for the time being. What is more, since the prototype itself does not represent a finished product that is ready to be released to the public, but is instead a functional implementation which helps exemplify how the proposed system can be used to the benefit of the software development community, its performance is not a major concern. Besides, as pointed out earlier, the comments-based suggestion approach might omit some of the most similar issues within the cluster itself, for example, if the comments they received were not upvoted or just did not have a high enough rating. This means that even though searching the entire dataset might be inefficient in a real world scenario where the collected information is of an enormous size, the direct distance comparison can still be utilized to present only the most similar issues within the cluster which were not shown but could still be of some value.

Incorporating the direct similarity comparison functionality itself was rather straightforward, since it did not involve assigning issues to a particular cluster or traversing the hierarchy tree. However, in order to avoid showing duplicate results, any comments that were already presented had to be taken into account. So, once the top 5 best rated comments are determined, the issues to which these comments belong are excluded from the distance comparison, which ensures that no issue would appear more than once in the results. Of course, as noted earlier, even though this appearance would be indirect when using the comments-based approach, from a user perspective, it is still redundant information. Again, as it was the case with the comments, the top 5 most similar issues are displayed to the user, as illustrated in Figure 5.10. As the figure shows, this section of the search results appears at the bottom of the page, right below the suggested comments (if any), with each entry containing the title of the issue as well as a link to the full issue thread.
It should be pointed out that while the comments need to have a minimum positive score of 1, if they are to be displayed, the issues have to pass a similarity threshold estimating how closely related they are to the original entry, as was the case with the similar issues within the assigned cluster, when utilizing the cluster-based approach. What is more, the threshold limit itself is the same as the one used for the comments, when the “high” precision option is selected. Besides, the result limit once again represents the maximum number of items that can be shown, meaning that if there are less than 5 issues which pass the predefined threshold, then the number of results is also lower, and if no issue is deemed to be similar enough, then nothing is returned to the end user.

Unfortunately, this implementation also suffers from the same shortcoming as the solution suggestions approach described in the previous section - only issues that have been posted at the time of the data collection are shown to the user. However, in this case, recency might be less crucial because if an issue is similar to a user’s query, it is most likely going to be helpful regardless of when it was originally posted. Either way, both predicaments could be addressed by periodically updating the data that is collected, for example, by re-collecting the issue comments or gathering more recent entries (and potentially discarding older ones).
6 System evaluation

This chapter presents the evaluation efforts that were performed in order to thoroughly assess the proposed solution. As mentioned in Section 3.5, the evaluation consisted of two separate stages - internal, done either by using computed measures or manually by the author; and external, which involved people from the outside, such as other researchers, students and practitioners in the software field. This sort of two-fold assessment ensured that the developed solution can be tested in both an objective and comprehensive way, meaning that the results from these evaluation sessions are less biased and therefore more reliable. Each section in this chapter presents a detailed account of the specific evaluation work that was conducted in order to assess a given aspect of the implementation, along with the results obtained from it.

6.1 Issue categorization

Once the clustering model was trained and the issues were categorized into separate clusters, the next step was to evaluate how well the model has managed to divide the dataset. The insight gained from such an evaluation can help determine whether the chosen approach presents a viable way for grouping distinct issue types based on their description and thus address RQ1. The evaluation itself was done through several internal measures as well as an exhaustive analysis performed by two independent groups of external reviewers. The details of each procedure are described in the following sections.

6.1.1 Internal evaluation

Evaluating the results of unsupervised clustering tasks has long been a compelling research topic. As previously mentioned, the main reason for this is the fact that, unlike supervised learning algorithms where a ground truth exists (i.e. the correct assignment of each data point within the set is known beforehand and thus can be used to establish the accuracy of the output), in unsupervised learning, such ground truth is not present. Therefore, in order to determine the quality of the clustering results, one cannot readily rely on such external measures but instead on different internal ones, which evaluate various characteristics of the output that are not related to the actual cluster assignments. As a result, there have been a number of studies exploring novel ways to address this problem, such as the work of Tibshirani and Walther (2005), among others. However, one of the most popular internal measures still in use today is the Silhouette coefficient (Guerra et al., 2012), which tries to estimate the separation between clusters, by calculating the mean distance between a given data point and all other points in the same cluster as well as the mean distance between the data point and all other points in the next nearest cluster. The values for this measure range from -1, which indicates incorrect clustering and 1 which signifies highly
dense, well-separated clustering, with a value around 0 illustrating that the clusters are overlapping.

Due to the fact that calculating the Silhouette coefficient required some heavy in-memory operations, which caused significant memory overhead, a random sample of the full data was used, instead of the entire collection. The sample consisted of around \( \frac{1}{3} \) of the total number of elements in the dataset, resulting in 9678 issue vectors. Nevertheless, having experimented with several differently sized, randomly generated samples, the outcome was very consistent, meaning that, at the very least, the metric was able to reliably measure the desired properties of the clustering, even if it did not evaluate every single data point in the set. In the end, the evaluation results showed that the Silhouette score of the trained model was 0.124 which indicates that the resulting clusters are overlapping. However, the fact that the coefficient relies on distance measures for its assessment which, as mentioned in Section 5.1.5, become meaningless in high dimensional space, means that the results of this metric cannot accurately represent the actual quality of the clusters. Nevertheless, the score serves to at least give some indication regarding the density of the clusters and their separation from each other, although, as mentioned, it should not necessarily be taken at face value. Moreover, this outcome can even be seen as an advantage, because, since the number of created clusters is greater than the number of desired issue categories, such an overlap can indicate that some clusters are closely related, hence they should probably receive the same category label.

Furthermore, these results are not particularly surprising, due to the fact that, as pointed out by Assent (2012), text documents often share terms among one another, and therefore achieving a good cluster separation is quite difficult. In this case, since entries have a number of identical attributes, the difference between similarity and dissimilarity among items becomes more ambiguous and as a result there is a certain amount of overlap, which might explain the aforementioned outcome. Moreover, since the categories used for classifying the various issues try to depict the underlying meaning of a given entry, while the clustering algorithm makes its assignment purely based on mathematical calculation, a perfect one-to-one mapping between the two is tough to accomplish, even though the very goal of all the activities leading up to the clustering is to create an objective representation of the dataset, based on mathematical models, which aim to capture the essence of the items within it.

### 6.1.2 Automatic cluster labeling

After the issues have been clustered, the next step is to assign descriptive labels to each one of the groups that were formed. Based on how difficult it is to choose a label for each cluster, it can be inferred whether the clusters are well-defined or not. Even though, as stated in Section 4.2.1, the final labeling will be done by experts, the labeling process itself can be conducted automatically without the need for human expertise. This means that, instead of people analyzing the clustering results and manually assigning the labels, they
are automatically computed based on the content of the cluster. This can provide an initial indication regarding the quality of the results and can aid further analytical efforts.

As mentioned by Manning, Raghavan and Schütze, there are two approaches when it comes to automatically assigning labels to the results of a clustering - differential and internal (2009, p.396). The differential approach chooses the labels for each cluster by comparing it with all the others and, through the use of statistical methods, determining which terms serve to uniquely characterize it in contrast to the rest. On the other hand, the internal technique assigns labels solely based on the content of the cluster, without taking into account any of the others. What is more, there are several options for choosing the label using this approach. One option is to use the titles of the documents which are positioned closest to the cluster center. The benefit of this method is the fact that titles are more readable and easier to comprehend for humans, which is an important concern if users are intended to interact with the clusters in some manner. However, as Manning, Raghavan and Schütze point out, “a single document is unlikely to be representative of all documents in a cluster” (2009, p.397). Another possible strategy is to use the highest weighted i.e. most relevant terms in the cluster centroid to form the labels, as done by Raja (2013). Even though, the resulting labels could be more representative of the cluster contents than a select number of titles, they are more difficult to process. As it can be seen, each approach has its pros and cons and so using a combination of several techniques can yield the most reliable results. Nevertheless, as Carmel, Roitman and Zwerdling point out, all automatic labeling methods have some inherent shortcomings (2009). For instance, even if the terms are conceptually related to one another, they could represent different aspects of the cluster topic. Furthermore, often times an appropriate cluster label might not even appear in the collection of terms. Last but not least, Carmel, Roitman and Zwerdling argue that the terms which are considered significant from a human-perspective are rarely seen as such from a purely statistical point-of-view (2009). Therefore, even if the automatic labeling approach is utilized, it should still be complemented by some form of manual human assessment in order to verify the suitability of the proposed terms and produce the most accurate labeling.

Since the benefits of the automatic cluster labeling approach far outweigh its flaws, it was decided to be employed as part of the research process. That is why, before the clusters were presented to the experts for evaluation, a sort of “pilot testing” was conducted in order to ensure that the clustering results are of sufficient quality and thus ready for evaluation in the first place. The testing consisted of analyzing all the clusters and manually assigning labels. The data used for the testing was acquired through a mixture of automatic labeling techniques, along with random samples of data from within each cluster. The goal of this approach was to create a more accurate and complete representation of the types of issues found inside the various clusters and thus facilitate the labeling itself.

As already mentioned, the differential labeling techniques rely on statistical methods commonly used in classification tasks where, as stated on several occasions, the target data
labels are known in advance. In that context, these methods are utilized for what is known as “feature selection”. Unlike feature extraction (described in Section 5.1.4), which was applied in order to extract the most relevant features from the text document dataset, feature selection aims to identify the features which have the greatest predictive potential and thus can be reliably used for assigning correct labels during the classification process. However, since clustering is an unsupervised task, it is unknown whether the resulting labels represent the ground truth. Therefore, in order to make use of the aforementioned statistical methods, the cluster assignments need to be presumptively considered as reflective of the ground truth, based on which, the predictive strength of a given feature can be estimated. It should be noted that, due to certain technological limitations, the approach that was employed for conducting this task was slightly different than the one proposed by Manning, Raghavan and Schütze (2009, p.396). Instead of taking into account the number of term occurrences within each cluster and then using it to determine which terms describe a given cluster in contrast to the others, the implementation uses the cluster assignments to identify the most influential terms across the entire dataset i.e. the terms which had the most significant impact in deciding the assignment of an issue. Each term (feature) receives a score based on its predictive capacity, calculated using the “mutual information” statistic, because as Manning, Raghavan and Schütze mention, its main alternative, the chi-squared measure tends to favor rare items, which are usually not a good indicator of cluster content (2009, p.278). Then, all terms that appear within a cluster are sorted according to their TF-IDF score (calculated during the feature extraction stage) and the terms which have the highest score and are deemed to be most influential according to their mutual information stats, are extracted. Even though this is not a typical differential cluster labeling implementation, conceptually it achieves a similar outcome, because due to the fact that the TF-IDF weight takes into account all terms within the dataset and the terms with a high score have the most considerable impact on the subsequent clustering, these terms tend to be different within each cluster.

Apart from the differential cluster labeling technique, the internal method was also utilized. As explained earlier, the internal labeling is done on a cluster-per-cluster basis, by selecting the terms positioned closest to the cluster centroid. Since terms with a high TF-IDF score are considered to be more relevant and thus have a more tangible influence on the cluster assignment, many of them are naturally positioned near the cluster center point. As a result, a portion of the terms extracted using both techniques, end up being the same. However, the two methods serve to complement each other, because the differential technique utilized here, can help counteract the phenomenon where a word appears to be important or descriptive but only because of its common co-occurrence along with another term, which is actually the one that carries significant meaning. For example, the term “open” might appear often in a small subset of issues and therefore receive a high TF-IDF score. Due to this high score, “open” is placed near the centroid of one of the resulting
6. System evaluation

clusters. However, the cause of this phenomenon is the fact that the term “file” is often used together with “open” and so “open” incorrectly receives a higher standing than deserved. Nevertheless, the word “open” is not limited to being used only in the context of files and so it appears in several other clusters as well. Therefore, it has a lower influence on the final cluster assignment. So, when utilizing the differential method, the word “file” will be extracted since it has both a high TF-IDF score and a high predictive potential, while “open” will be rightfully ignored. On the other hand, the internal labeling method, could help discover terms that did not necessarily receive a high TF-IDF score but are nevertheless relevant in the cluster’s context, thus once again demonstrating how the two techniques can complement one another.

On top of the terms acquired through the aforementioned techniques, a random sample of terms (which have not yet been selected) was also extracted, because such a sample would serve to provide a more complete representation of the overall composition of the cluster. A total of 10 terms were gathered through each one of the three techniques. Table 6.1 shows an example of the output for several of the clusters.

Table 6.1: A sample of the terms contained within some of the resulting clusters, extracted using three separate approaches – differential, internal and random (the bold font indicates words gathered through multiple techniques)

<table>
<thead>
<tr>
<th>Differential</th>
<th>Internal</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>route, path, nest, handle, dynamic, transition, match, hook, longer, index</td>
<td>route, path, nest, transition, incorrect, url, webpack, match, child, handle</td>
<td>call, segment, warn, override, rail, abort, refresh, gulp, fail execute, carousel</td>
</tr>
<tr>
<td>return, instead, null, always, promise, empty, even, true, query, false</td>
<td>return, instead, null, promise, true, result, string, false, throw, empty</td>
<td>original, handle, var, mode, resolution, tag, beta, delay, sub, hidden</td>
</tr>
<tr>
<td>test, case, bonfire, write, integration, incorrectly, setting, print, investigate, since</td>
<td>test, case, run test, run, fail, test case, test fail, pass, bonfire, challenge</td>
<td>setup, look, translation, correctly, correct, properly, end, use npm, development, request</td>
</tr>
</tbody>
</table>

As the data illustrates, there is a noticeable overlap between the differential and internal terms, denoted in Table 6.1 through the use of a bolded font, while the random terms do not really indicate a common pattern. However, some of the randomly selected words still fit with the overall context of the cluster. For instance, the differential and internal terms in the first row imply that the cluster is related to URL routing, a notion also reflected by some of the random terms such as “refresh” and “override”. The same applies to the last row in Table 6.1, which is clearly related to testing, with some of the random words also pointing in that direction to some extent (e.g. “correctly”, “properly”). All in all, the term analysis revealed that if a cluster is well-defined, the terms within it tend to give a decent indication regarding its main topic.

Nevertheless, as noted earlier, a list of terms is inherently more difficult to comprehend
and examine than a full phrase and so the analysis task was further complemented by extracting the full titles of several issues from each cluster. Again, this was accomplished through the use of both an internal approach, which selected the 5 closest documents to each cluster center, as well as the random sample approach, which gathered the titles of 5 randomly chosen issues, different from the ones already selected through the previous method. As Table 6.2 illustrates, the titles served to provide further proof about the subject matter of the different clusters.

Table 6.2: A sample of the full-length titles of the issues inside some of the clusters, extracted using an internal and a random approach (and separated from one another by short horizontal lines)

<table>
<thead>
<tr>
<th>Titles of closest docs</th>
<th>Random titles</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to receive a oauth token with vue since the routes are behind #</td>
<td>Proposal: fast path for assert.deepEqual with typed arrays</td>
</tr>
<tr>
<td>The route /:path(abc—xyz)* doesn’t work.</td>
<td>tooltipTemplate equivalent in Chart.js 2.0 ?</td>
</tr>
<tr>
<td>v4: Should paths merge in child routes?</td>
<td>Question - have nested routes ignore browser back/forward buttons?</td>
</tr>
<tr>
<td>How to use ‘-’ and other symbols in route path</td>
<td>How to handle direct routes using Express.js?</td>
</tr>
<tr>
<td>cannot route batch path /classes/,Installation</td>
<td>Enhancement Request: Automatic blueprint routes for ‘binary’ data</td>
</tr>
<tr>
<td>Why moment(1000<em>60</em>60).format(‘HH:mm:ss’) returns ”02:00:00” instead of ”01:00:00”?</td>
<td>Collection.findOne(id) returns wrong results when id is undefined</td>
</tr>
<tr>
<td>‘Ember.A(m)(null)’ now returns ‘null’ instead of ‘[]’</td>
<td>morphTargets : true does not work on THREE.ShaderMaterial</td>
</tr>
<tr>
<td>dirty returns always true</td>
<td>baseUrl return nothing</td>
</tr>
<tr>
<td>findOne (and its derivatives) must return ‘undefined’ instead of ‘null’ (for ES6)</td>
<td>_.every returns true for empty or undefined collections</td>
</tr>
<tr>
<td>$(‘meta[name=”description”]’).attr(“content”) returns empty</td>
<td>Quill not returning html entities</td>
</tr>
<tr>
<td>tests using consistent logic?</td>
<td>Description and tests don’t match</td>
</tr>
<tr>
<td>Clean up tests etc, now that should.js has been upgraded</td>
<td>Testing the issues bot</td>
</tr>
<tr>
<td>Impossible to write a protractor test once you use a toolbar (because there is a $timeout every 5 seconds)</td>
<td>Testing a new package doesn’t work (wrong path)</td>
</tr>
<tr>
<td>Assertion errors caught in next test when mocha timeout is applied</td>
<td>Test check registry status?</td>
</tr>
<tr>
<td>Clean up the tests folder</td>
<td>Deferred tests fail in Android 4.0</td>
</tr>
</tbody>
</table>

As the issue titles show, the first and last row also support the assumption that the clusters represent issues regarding “URL routing” and “Testing” respectively, while the
titles in the second row reinforce the suspicion that the cluster’s subject is method/function return values. Moreover, even the randomly selected titles are strongly related to the overall cluster topic which signifies that the issues contained within these clusters, deal with a rather specific problem.

6.1.3 Pilot testing

As already mentioned, both the terms and the titles were used for facilitating the cluster labeling process which was performed as part of a pilot testing session. The goal of the session was to serve as a pre-evaluation step which could help verify the quality of the clustering results before presenting the data to the experts for a more comprehensive and in-depth examination.

The data showed in Table 6.1 and Table 6.2 represents only a small sample of the data\textsuperscript{22} that was used for the analysis, but it still manages to serve as an accurate depiction of the overarching composition of the clustering results. In fact, the initial examination revealed that the majority of the clusters had a clear thematic pattern, with the issues contained within them, addressing a particular type of problem. Furthermore, around $\frac{2}{3}$ of the clusters (i.e. 20 out of the 30) had an easily distinguishable main topic which could be identified by briefly examining the terms and titles that were extracted. This means that all the processing activities done prior to the model training, did in fact contribute to representing the similarity between text excerpts in a mathematical form, which is in itself a significant achievement. Of course, despite the rigorous data preparation process, some items ended up being placed in the wrong cluster. Nevertheless, this problem may be solved by further refining and improving the different data processing components as well as the configuration of the clustering model.

6.1.4 Expert analysis

After the pilot testing was complete, the next step in the evaluation process was the expert analysis, the purpose of which was to serve as an external, unbiased assessment of the issue clustering results. The task of the experts was to examine a sample of the clustering data and try, to the best of their abilities, to assign a category label to each cluster, based on the terms and titles contained within it. While the reviewers were asked to analyze the exact same data that was used during the pilot testing, they were only presented with a subset of the total number of clusters, more specifically the majority of the clusters which were deemed to be most well-formed as a result of the previous evaluation step, meaning a total of 15 clusters. Since the proposed solution is an initial proof-of-concept implementation, having only a portion of clusters that are well-defined is more than enough because it demonstrates the feasibility of the chosen approach and serves as evidence to the potential

\textsuperscript{22} All data can be found here - https://drive.google.com/open?id=0BwdN1KiwhxCtakKlLUdpZXiNHc
of the system to identify distinct issue categories based on natural language. Of course, in order to improve the accuracy of the solution and ensure that all of the resulting clusters have an easily discernible common theme, additional refinement would be required. What is more, filtering through the data that is presented to the experts, makes sure that they are not overwhelmed by having to analyze an excessive amount of information and that the analysis sessions will be as productive as possible, because the reviewers do not have to waste time and effort on examining clusters which are visibly cluttered and chaotic. Last but not least, if the experts managed to come to a consensus and mutually agree upon a set of descriptive labels for each of the clusters that were presented to them, it would help to further support the claim that the prototype was able to separate the issue dataset into distinct categories, thus validating its feasibility and, in turn, providing an answer to RQ1. The expert analysis itself involved two different groups made up of three external reviewers each. The first group consisted of fellow researchers, while the second included practitioners in a number of software related fields. The fact that there were three people in each group, allowed them to have rich discussions among each other, while still offering them the opportunity to participate in all conversations. What is more, all of the reviewers had sufficient experience working with JavaScript, meaning that they would be able to provide relevant feedback regarding the examined data. The evaluation procedure consisted of two phases. During the first phase, the participants had to individually analyze the clustering data and assign a label (or multiple labels) to each category. This was followed by a group review session, where the experts could discuss their assignments and try to agree on a final label for each cluster. In order to determine whether the participants were confident in their selection, they were asked to rate each chosen label on two accounts - how well it represented the data within the cluster as a whole and how well it described each individual instance. The rationale here was that these questions could help determine if the experts were certain that the resulting clusters not only have a clearly distinguishable main theme but also that they are self-contained, meaning that there is a distinct issue type within each one. The group session was concluded with a short semi-structured discussion which aimed to gather some additional feedback from the experts about the clustering results, the developed solution and the research topic itself. A detailed description of the exact procedure that was followed during the expert analysis sessions as well as justifications for the various design choices that were made, can be found in Appendix C.

The results of the first phase of the expert analysis (i.e. the individual cluster labeling), showed that there was a considerable consensus among the participants in both groups. Furthermore, the labels they used were often closely related and sometimes even overlapping with the classification made during the pilot testing of the clustering data, described in Section 6.1.3. Even though this outcome appeared to be quite promising, the follow-up group session needed to be conducted, before making any conclusions. As mentioned, the main part of the group session was meant for the experts to review their assignments and
come to an agreement about a final labeling for each of the categories. As it turned out, due to the fact that they used very similar terms to describe a large portion of the clusters, agreeing on a final label was not particularly difficult. What is more, the experts had a high confidence about the chosen labels for these same clusters, meaning that they were quite certain that their choice was descriptive of the data within the given cluster. However, there were still some clusters that caused a significant debate among the participants, which consequently lead to lower confidence levels regarding the eventual label that was selected. This might be due to the fact that the aforementioned clusters were not sufficiently well-formed and as a result, contain issues which are not similar enough to be labeled using the same term. Alternatively, this could also mean that the label, which the experts agreed on, was not the most descriptive one that could have been used (or even both). Nevertheless, the participants in the first group were able to come to a consensus about all of the cluster labels and had a high confidence regarding over $\frac{2}{3}$ of their choices. Therefore, it can be concluded that a considerable portion of the clusters resulting from the model training, not only have a clearly discernible theme but also contain distinct types of issues, meaning that the clustering model was able to separate them into self-contained groups, thus providing strong evidence towards answering RQ1. In fact, the labeling results attained during the second expert analysis session further corroborated the outcomes of the first one, which adds even more weight to the claim that natural language processing and machine learning can be effectively utilized to categorize distinct types of programming issues based on their textual description. Moreover, since the additional review sessions have been conducted by another researcher adhering to the study protocol explained in Appendix C, this serves to show that the protocol has been well-designed and as a result, the study can be properly reproduced. For a more detailed presentation of the session results, please refer to Appendix D.4.

Finally, after the experts were done reviewing their label assignments, they were engaged in a short discussion, in order to get their opinion on a variety of subjects concerning the research. First of all, when asked if they had any trouble completing their task during the individual labeling part of the analysis, they admitted that they faced significant difficulties, due to the fact that the data appeared to be rather mixed - some portions of it did not contribute in any way whatsoever to finding an appropriate label and sometimes even caused more confusion. As a result, they could not easily identify a common theme within the clusters, without having to examine the data multiple times. Nevertheless, the experts all agreed that the differential terms were the most helpful for deciding on a label, closely followed by the full issue titles. This lead to a discussion regarding the trade-off of using terms and titles. While terms can help get a broader sense of the content within the cluster, they do not provide any context. On the other hand, titles provide plenty of context but

23 Their assignments can be seen here -

https://drive.google.com/open?id=OBwdNIKiwHXUcFVPhpWnhLVU
only represent a small subset of the data. These same concerns were of course mentioned in Section 6.1.2 and were taken into account when choosing how the data should be displayed in order to facilitate the labeling process. Overall, the participants thought that the research topic is very interesting and that it has significant potential. However, they also echoed some of the author’s concerns regarding the design of the solution, such as the fact that the categorization is highly dependent on the way the issue title is formulated. Furthermore, since the people who are submitting the issues are not necessarily experienced in the subject domain, they might use the wrong terminology to describe their inquiry, thus exacerbating the problem even further. Another problem that they saw was that the resulting categories seemed to have a different level of specificity, with some clusters concerning a more broad area, while others being about a very particular topic. The experts believed that these issues should be addressed if the proposed solution is to have any real world applicability. Nevertheless, they thought that, if it is able to work well, the solution could be utilized in many different domains and can be especially helpful in facilitating the process of automatic tagging on platforms like forums and Q&A websites. Other potential applications that were mentioned included issue triaging, automatic issue identification and preliminary problem resolution (see Appendix D.4 for more information). Lastly, the experts were asked to share their opinion about the evaluation itself and the way it was planned and executed. For the most part, none of them had any complaints regarding the overall procedure, with the exception of the initial description of the task and their specific responsibilities. The reviewers felt that the goal of the task could have been better communicated, which would have made it easier to do what was expected of them.

The expert feedback gathered through the post-review discussions provided an extremely useful and critical outside perspective on the developed solution as well as its strengths and weaknesses. What is more, a lot of the comments of the participants confirmed previous concerns that were raised throughout the development process and should serve as a starting point for further investigation and future improvements of the system as a whole. However, these results also illustrate the fact that the categorization has proven to be useful from the perspective of both researchers and practitioners alike, thus demonstrating the relevance of the proposed solution. For full details of the results and feedback acquired through the expert analysis sessions, see Appendix D.

### 6.2 Issue assignment

The pilot testing of the clustering data and the subsequent expert analysis sessions helped determine that the clustering model was in fact able to divide the training data into separate categories based on the type of the issues, thus providing strong supporting evidence towards answering RQ1. However, despite all of these efforts, it was still uncertain if the model could accurately assign new and unfamiliar issues to the appropriate category, as stated in RQ2. That is why, in order to address this matter, additional evaluation measures had to be
6. System evaluation

6.2.1 Evaluation approach

One possible evaluation approach would have been to split the original dataset into a training and test set, for instance, using a 80:20 ratio, then creating a clustering model with the training set and finally, testing the accuracy of the model with the test set. However, this scenario requires manually analyzing and labeling thousands of issues in order to establish a ground truth which could then be used to determine if the cluster assignment is correct. The problem with this approach is that it essentially renders one of the most significant contributions of this study, namely harnessing the power of unsupervised machine learning to automate an extremely time-consuming and labor-intensive process such as issue categorization, completely meaningless. Nevertheless, an alternative strategy, which is often utilized by researchers for evaluating classification models, shown for example in the work of Cohen and Hersh (2005), is the so-called “gold standard” classification. Using this approach, a small number of items are first analyzed by a human expert and labeled accordingly. Then, the labeled items are introduced to the classification algorithm, which assigns its own labels to the collection. Finally, the classification accuracy is estimated by comparing the assignments given by the human expert and by the algorithm. The benefit of this strategy is the fact that it avoids the aforementioned problem of having to analyze an excessive amount of data, while still providing a reliable indication regarding the accuracy of the trained model. Even though this method is primarily used for evaluating classification algorithms, it can still be applied to a clustering model, because when using pre-labeled data, the clustering becomes supervised, which is essentially a form of classification. However, since only a subset of the clusters have been formally analyzed by external experts and given a specific label, all issues that are introduced to the model should fall within these categories. The reason for this is that without an accepted label, the assignment would have to be done using the cluster index, which makes the whole operation lose its purpose, because the main rationale behind the evaluation is to determine if the model can be useful in a real world scenario, where items receive appropriate and descriptive labels instead of numerical indices that do not carry any meaning whatsoever.

6.2.2 Data collection

In order to evaluate the assignment accuracy of the clustering, first a new dataset of issues had to be collected. Moreover, these issues had to come from repositories which had not been considered during the data collection stage so as to ensure that the data would be completely foreign to the model and thus help to more objectively determine how well it is able to perform its function. For this purpose, a collection of issues was gathered from JavaScript repositories which had between 5000 and 10000 stars i.e. repos which have not
been examined earlier, but still popular enough so that they meet certain quality criteria, while also avoiding the potential problems mentioned by Kalliamvakou et al. (2014). At the time of the collection, there were 325 such repositories\(^{24}\), from which 50 projects were selected at random. Then, for each of the chosen repositories, all issues that met the same requirements as the ones established during the initial collection were retrieved, from which a single issue was selected, again on a completely random basis. However, since the repositories were chosen by chance, there is no guarantee that any of the issues within them meet these requirements, meaning that it is difficult to specify an exact number of issues that are to be extracted. Furthermore, the titles of the collected issues might not contain any of the terms found in the extracted feature vocabulary and as a result, they would be represented as null vectors, thus making them unfit for assessing the issue classification process. Nevertheless, these problems could be alleviated by defining a slightly larger number of repositories and/or issues than the amount which is desired to be gathered, which would serve to effectively compensate for such hindrances. It should be noted that in the case when a randomly chosen repo does not contain any issues that meet the collection requirements, the repo is simply ignored. On the other hand, in order to determine whether the title of a chosen issue would be represented as a null vector, the issue has to be gathered and then put through the vectorization process described in Section 5.2.1 and if the resulting vector contains solely null values, the entire issue is removed from the dataset. Last but not least, even if an issue manages to pass all of the aforementioned tests, the chance that it will fall into one of the clusters which have been given a label, is roughly around 50% since they represent half of the total number of clusters that were formed during the model training. Therefore, taking all of these considerations into account, collecting around 50 random issues should produce between 15 to 25 entries that could be used for evaluating the model. What is more, this entire process was repeated three times in order to gather three separate datasets which could then be individually labeled and submitted to the clustering model. That way, there will be multiple, independent samples of clustering assignments which can be compared so as to more reliably assess how well the trained model can estimate the correct category for a given issue.

The data gathering stage resulted in three datasets containing 44, 47 and 47 issues respectively. After the collection, the data was analyzed and all issues that fell into one of the labeled categories were coded with an appropriate label. However, the labeling process itself suffered from a certain degree of bias since, as mentioned earlier, it was conducted by the author. What is more, due to the fact that analyzing the clustering results during the earlier stages of the research meant that the author was aware which keywords seemed to have the most significant impact on the cluster assignment, this information could have had an additional impact on the labels that were given. Therefore, this part of the evaluation process should be repeated once again with the help of external reviewers in order to obtain

\(^{24}\)https://api.github.com/search/repositories?q=language:javascript+stars:5000..10000
more reliable results. Nevertheless, even in its current format, this analysis can produce valuable insight which could serve as a starting point for future research efforts.

6.2.3 Evaluation results

As a result of the labeling process, the total number of issues was reduced to 20 for the first dataset, 17 for the second and 19 for the third. The evaluation itself was done through the use of a variety of statistical measures which aimed to help assess the accuracy of the machine learning model with a higher degree of certainty even when using a relatively small amount of test data. Table 6.3 shows a summary of the assignment results for each of the three evaluation datasets, along with the various statistical measures that were calculated as well as their mean and standard deviation values. It should be noted that the value range for each metric is either between -1 and 1 or 0 and 1, with higher values signifying a more favorable score. Furthermore, the standard deviation was quite low for each measure, meaning that the results were consistent across all datasets.

Table 6.3: A summary of the results acquired through several statistical measures, aimed at evaluating the issue assignment capabilities of the clustering model

<table>
<thead>
<tr>
<th>Metric</th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total issues</td>
<td>44</td>
<td>47</td>
<td>47</td>
<td>46</td>
<td>N/A</td>
</tr>
<tr>
<td>Labeled issues</td>
<td>20</td>
<td>17</td>
<td>19</td>
<td>19</td>
<td>N/A</td>
</tr>
<tr>
<td>Accuracy [0, 1]</td>
<td>0.70</td>
<td>0.65</td>
<td>0.63</td>
<td>0.66</td>
<td>0.04</td>
</tr>
<tr>
<td>Adjusted Rand index [-1, 1]</td>
<td>0.54</td>
<td>0.44</td>
<td>0.59</td>
<td>0.52</td>
<td>0.08</td>
</tr>
<tr>
<td>Homogeneity [0, 1]</td>
<td>1.00</td>
<td>0.93</td>
<td>0.96</td>
<td>0.96</td>
<td>0.04</td>
</tr>
<tr>
<td>Completeness [0, 1]</td>
<td>0.78</td>
<td>0.85</td>
<td>0.82</td>
<td>0.82</td>
<td>0.04</td>
</tr>
<tr>
<td>V-measure [0, 1]</td>
<td>0.88</td>
<td>0.89</td>
<td>0.89</td>
<td>0.89</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The most straightforward metric is of course the accuracy, which simply shows how many of the issues received a correct label assignment. As Table 6.3 illustrates, the model has a very respectable accuracy of around 66% meaning that 2/3 of the issues were assigned correctly. This is a very promising outcome at such an early stage in the development of the proposed solution and serves to validate the feasibility of the chosen approach. However,  

---

25 All issues used for the evaluation, along with their corresponding labels can be found here - https://drive.google.com/open?id=0BwdNIKiwHxCtd180ZE5WUN4T28
there is a multitude of other metrics which can reveal more in-depth insights about the assignment results. One such measure is the adjusted Rand index (Guerra et al., 2012), which computes the similarity between the “true” and the predicted clustering, by taking into account each pair of samples and counting the pairs assigned in the same or in a differing manner. Furthermore, since this metric is also normalized for chance, it can be used to verify that the assignments were not random (if the value is close to 0), even if the analyzed sample is of a very small size. In this case, the adjusted Rand index was around 0.5, thus indicating that there is in fact a discernible pattern in the cluster assignments.

The next set of evaluation criteria are very closely related to one another. The first, homogeneity, also often referred to as “purity”, signifies if each cluster contains only members of a single class. On the other hand, the completeness measure indicates whether all data points belonging to a single class are only assigned to a single cluster. Finally, V-measure, an evaluation criteria proposed by Rosenberg and Hirschberg (2007) represents the mean of the homogeneity and the completeness scores. As the data shows, the scores for each measure are quite high, especially for the homogeneity, which serves to illustrate that the clustering model can accurately assign new issues to the specific category where they belong. However, the drawback of these measures is that they are not normalized against chance, meaning that a completely random assignment will not always yield the same results, especially when the number of clusters is rather big (over 10). Therefore, these criteria may be somewhat unreliable in this context, which is one of the reasons for complementing them with the adjusted Rand index value, which addresses this particular problem and, as mentioned, demonstrates that the clustering assignment results are not coincidental.

### 6.3 Solution suggestions

The last part of the evaluation efforts involved testing the web prototype that was developed. The purpose of the assessment was to determine whether the implementation can in fact help users find relevant solutions to their programming issues and thus provide an answer to RQ3. Again, the evaluation consisted of two separate stages - an initial pilot testing analysis, conducted by the author, followed by several user testing sessions conducted with people representative of the main target audience which can mostly benefit from such a solution. The following sections present both the specific procedure that was adopted during each evaluation stage, along with the results obtained from it.

#### 6.3.1 Pilot testing

Before conducting the prototype evaluation with actual users, a pre-evaluation activity was performed, similar to the one done prior to presenting the clustering data to the external reviewers. The purpose of this pilot testing was to determine whether the results produced
by the application are relevant to the submitted search query and/or useful in some way. Furthermore, the session also aimed to examine how does the output change depending on the currently selected “precision” option. The testing itself was done by utilizing different issue titles as search queries and assessing the results. The issues themselves were once again randomly selected from JavaScript repositories which had between 5000 and 10000 stars. The final evaluation dataset consisted of 16 issues in total - each collected from a different randomly chosen repo. Table 6.4 presents a sample of the issue titles used for assessing the system.

**Table 6.4:** A sample of the issue titles used for conducting the initial pilot testing of the web prototype

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>How can I change the background color opacity?</td>
</tr>
<tr>
<td>return_buffers with pub/sub is broken since 2.2.0, detect_buffers - since 2.0.0</td>
</tr>
<tr>
<td>Bower version ?</td>
</tr>
</tbody>
</table>

Even though the evaluation itself was rather informal, it proved to be very helpful for determining how well does the prototype work. The analysis yielded a number of observations regarding the functionality and general behavior of the implementation, which together with the feedback received during the actual user evaluation sessions, can aid in perfecting the proposed solution. One such observation was the fact that quite often, the majority (sometimes even all) of the comment suggestions came from the same issue. If the issue itself is pertinent to the search query then this could give users a quick overview of the most useful comments that have been posted on a given thread, without them needing to browse through it. However, if it is not, that would prevent other potentially relevant answers from appearing in the results. Therefore, in the future, the implementation could be adjusted so that only the top 1 or 2 comments from a single issue are shown, in order to balance out the suggestions.

Another phenomenon that made a strong impression was the specific words in the search query which had the biggest impact on the final output. Based on both the comments and the similar issue suggestions, it appeared that often times, words which, despite carrying some meaning were still not very descriptive, had a considerable influence on the results. For example, terms such as “loading”, “broken”, “version”, “add” and so on, which could be very useful in combination with other words, but on their own provide far less information regarding the context of the entry, most likely played a key role in determining the cluster assignment of an issue and consequently the suggestions shown to the user. This means that these terms were probably given a high TF-IDF score during the feature extraction.

---

26 The collection of issue titles can be found here - [https://drive.google.com/open?id=0BwdNIKiwHxCtQ054Rj1MbW1rOVU](https://drive.google.com/open?id=0BwdNIKiwHxCtQ054Rj1MbW1rOVU)
stage and as a result carried a lot of weight during the subsequent clustering. On the other hand, genuinely important words like names of specific technologies such as “jquery”, “vue”, “bower”, etc., did not have as notable of an effect, especially when combined with the aforementioned terms. The conclusion that can be drawn from this, is that the feature weights have to be recalibrated so that they can better reflect the importance of the different terms within the dataset. For instance, names of functions, projects, etc. should have a higher feature weight, because, due to their specificity, they convey a lot of information regarding the nature of the issue, while common, yet somewhat vague words, should receive a slightly lower weight, unless they are used as a part of a multigram expression. Unfortunately, this problem is further exacerbated by the fact that the names of some libraries and frameworks represent actual words found in the English language (e.g. “react”, “express”, “backbone”), which makes it even more difficult to determine whether the entry concerns the technology itself or if it just contains the word used in its traditional context.

Despite the aforementioned shortcomings, the main components of the prototype performed quite well, especially considering the fact that this is the very first version of the implementation. For example, the similar issue suggestions often appeared to be relevant to the search query, unless one of the highly weighted non-descriptive words mentioned in the previous paragraph was affecting the results. The suggestion precision range slider was also working as expected, either broadening or narrowing the scope of the results, depending on the currently chosen setting. In fact, on several occasions, it appeared to be functioning even better than expected because, when the precision value was decreased, the suggestions seemed to reflect the wider context of the key terms used in the query. On the other hand, when the precision was set to “high” that often resulted in no suggestions being shown, which is probably caused by the fact that the dataset is not that large and so when the similarity threshold is quite low, a lot of times there are not many issues that fit the criteria. Therefore, the default precision setting (the one between “high” and “moderate”) presents a reasonable option that could yield satisfactory results for most searches. Lastly, the comment suggestions, which are the centerpiece of the prototype, did not usually provide a direct answer to the search query, which is likely due to the fact that since the issues posted on GitHub concern a very specific technology, the answers are also rather specific. However, the relation between the search and the answers that were being displayed was often times quite apparent and so the results could still be useful for the end users.

The pilot testing of the web prototype revealed a lot about how well it performs its primary functions. It helped uncover both some of its shortcomings as well as some of its strengths. However, it is important to note that since the author was not aware of what the actual solutions to the issues used for the test were, the judgement of whether a specific suggestion is relevant or not was rather subjective. Furthermore, no piece of software can be thoroughly evaluated without being presented to the people who are eventually going to be using it. Therefore, the feedback gathered during the user evaluation should be taken into account.
consideration before making any definitive conclusions. Moreover, the evaluation sessions would help examine the prototype much more in-depth, because users will be able to explore and see not only if the suggestions that are directly provided to them are of use, but also if the implementation can help them acquire new knowledge or bring them closer to finding a solution to their inquiries.

6.3.2 User evaluation

Once all aspects of the system had been developed, the next logical step was to illustrate how software developers and programming enthusiasts alike could benefit from its capabilities. For this purpose, the web prototype that was created, was to be tested by users who represent the main target group i.e. the people whom the application aims to assist. The primary goal of this evaluation was to examine whether the prototype can in fact help users find solutions to their programming issues in an easier and more efficient way as well as whether it can be helpful to both seasoned professionals and relative novices.

The implementation itself was initially intended to be hosted on OpenShift\textsuperscript{27}, which is a platform-as-a-service (PaaS) solution that supports a wide range of web technologies and offers a free tier, which allows developers to deploy their applications in the cloud. Apart from the purely practical considerations that guided this choice, OpenShift, and cloud platforms in general, present an ideal environment for this type of system due to their decentralized architecture and the considerable processing capacity that is on offer. Since the proposed solution is intended to work with truly Big Data, scalability is of utmost importance and therefore the application can benefit immensely from being hosted on such a platform, not only as the data being analyzed and the potential traffic increases, but even at its current state. However, as it turned out, the application was using too much of the resources allowed by the service, such as CPU and memory, and as a result, it kept getting stopped automatically after being used for a short period of time, essentially making it unusable. That is why, an alternative approach for sharing the prototype with prospective users was adopted. Since the application was running perfectly on the local machine used during the development process, the solution that was chosen was to use an external service, which created a temporary secure proxy to the local server that could be accessed through the web. This allowed for the prototype to be running on the local machine, while at the same time being accessible to users anywhere in the world, as long as they were connected to the Internet. As previously illustrated in Figure 5.4, the application looked a lot like a traditional search engine. Of course, the aesthetics and usability of the interface were not very refined, first of all because these aspects were not the subject of the evaluation but also because the main purpose of the prototype itself was to provide users with a platform through which they could take advantage of the underlying functionality in order to address

\textsuperscript{27}https://www.openshift.com/
a specific need. What is more, some of the initial tests of the application showed that its performance would also require further optimization. However, since the implementation was running locally, on a machine that had rather limited resources, it would need to be migrated to a production environment, with its own dedicated server, in order to get a more accurate idea of how it would behave in a real world scenario.

The user evaluation itself consisted of two main stages - one pre-session activity that users had to complete prior to the testing, followed by the actual test procedure. During the initial activity, participants were asked to describe in a sentence, each of the 5 most recent JavaScript issues that they had to deal with and then send that information to the researcher. The issues could be of any nature and could be about any library or framework as long as they concerned the JavaScript programming language. The purpose of this activity was to collect data, which would later be used for the evaluation, meaning that users would try to find a solution to each one of the 5 issues that they described. The advantage of this approach is that, the researcher does not influence the outcome of the test in any way as well as the fact that since users have, more than likely, already solved all of these problems, they would know when they see a suitable solution and thus be more objective in their assessment of the prototype. The second stage of the evaluation was further divided into three parts - a pre-test questionnaire, the actual prototype testing and a post-test questionnaire. The pre-test survey aimed to gather more information about the background of the participants, their level of experience with JavaScript and general search tendencies - data which could provide additional context to their feedback later on and thus help to better understand it and ensure that the correct conclusions were being made. As mentioned, the testing of the prototype consisted of users trying to find solutions to the 5 issues which they themselves identified before the session, through the use of the web application. If they managed to do so, participants were asked to rate the answer they discovered on a scale from 1 to 10, based on its relevance to their search query and overall completeness. Once users find a solution to an issue or give up on their efforts to do so, they move on to the next one, and this process is repeated until there are no more issues left. Before the start of the session, participants were also asked to share their screens so that the researcher could observe how they interact with the prototype during the test and take notes. Furthermore, the study subjects were encouraged to share their opinions and comments on the application as they were using it. Finally, after the testing, users were presented with another short survey which was intended to gather their feedback regarding various aspects of the prototype as well as the whole idea that it served to represent. A detailed description of the exact study protocol which guided the evaluation, can be found in Appendix E.

In the end, the user evaluation was conducted with a total of 6 participants who embody the types of people to whom such a solution would be most beneficial - developers, students and programming enthusiasts who are often faced with all kinds of software related issues
and so facilitating the process of finding relevant answers would make a real difference for them. The pre-test questionnaire showed that the chosen participants were people who have had experience with web development and JavaScript as a whole and even though they might not describe themselves as experts on the subject, their feedback is still relevant and reliable. Furthermore, when searching for solutions online, they mostly rely on the help of search engines, Q&A website like StackOverflow and reference platforms such as w3schools\textsuperscript{28} or MDN\textsuperscript{29}. Even though all users admitted to using GitHub for finding solutions to their programming related issues, their general opinion of the platform is not as high as it is for the aforementioned resources. This is clearly reflected in their answers, which stated that it is somewhat difficult for them to find the information they are looking for on GitHub and, naturally as a result they declare that they do not find the platform to be as helpful as a site like StackOverflow and consequently do not use it as often. Unfortunately, these previous experiences could serve to bias their assessment of the prototype because they might have already made up their mind that GitHub is not a good source for finding solutions to programming issues - a point that should be considered in the subsequent analysis of the evaluation results. Another interesting outcome of the pre-test questionnaire was the fact that users expressed a strong preference for answers containing code samples compared to other responses that consist of detailed explanations or links to external resources. This insight could be essential for future iterations of the prototype and could point to a specific improvement that may be implemented in order to make the application more useful to the end users.

Unfortunately, the actual prototype testing demonstrated that, at least at its current form, the implementation was not very effective in helping people find solutions to their programming problems. Out of the 6 participants involved in the study, 5 of them were able to discover some sort of an answer to a given query, but out of the 30 total searches that were made, the application produced a relevant answer for only 9 of them, which constitutes less than $\frac{1}{3}$ of all cases. Furthermore, whenever they were able to find an answer, users rated it with 5/10 on average, due to the fact that, according to them, despite being related to their query, it was still much too general and did not provide a specific solution to their problem. Nevertheless, users admitted that the answer they were able to discover, would be a good starting point for finding a definitive solution. Participants also found it frustrating and somewhat confusing that often times it appeared that the application did not take certain keywords into account, when displaying the results. Therefore, it was quite difficult for them to find what they were looking for and even the times when they managed to reach an answer, this took a considerable amount of time and effort and required that they changed or adjusted their original query on a number of occasions. What is more, during the session, several of the participants shared their belief that one would not be able to

\textsuperscript{28}\url{https://www.w3schools.com/}
\textsuperscript{29}\url{https://developer.mozilla.org/en-US/}
find a solution to a problem related to “vanilla” JavaScript or just a more general and abstract inquiry on GitHub, due to the fact that since each project on the platform has its own dedicated “Issues” section, the issues found there are overwhelmingly related to a specific library or framework. However, even though some of the issues that they used for the search were indeed quite generic, which could have explained the somewhat poor results that they received, a good portion of the issues concerned a particular JavaScript project but yet did not seem to produce output that was noticeably better. Nevertheless, the main reason for this negative outcome is most likely the fact that the solution did not have a sufficient amount of data to work with. Since the prototype was only searching for answers within a limited set of repositories, many queries were bound to return no relevant results, especially the ones concerning libraries or frameworks which had not been collected during the initial data gathering stage.

Naturally, given the outcome of the evaluation sessions, the post-test feedback received from the participants was not particularly positive. Not only did they not think that the prototype provided them with relevant results, they were also unable to find answers to the majority of their questions and viewed the search process itself as challenging. However, many of the study subjects did confess that, despite all of this, they were still able to discover something interesting and learn something new, which demonstrates that the prototype did in fact manage to provide some value to the users, even if it did not meet its primary goal. This, coupled with the fact that, as the testing sessions showed, around 1/3 of the time, users were able to come upon an answer, serves to indicate that, despite the early stage of its development and its apparent shortcomings, the prototype can still provide useful information. Moreover, the participants seemed to agree with such a statement, because when asked if this type of application would be helpful for finding solutions to programming issues, they had a largely positive response. This means that, even though they might have been somewhat disappointed by the execution, they still see the potential of the idea itself. Nevertheless, as their feedback also showed, the prototype cannot compare to the traditional sources of information on which they rely, which as the pre-test survey helped identify are mainly Q&A sites and reference platforms. Lastly, users were asked to assess specific aspects of the prototype and their impact on the output. For example, opinions were divided about the impact of the “precision” setting, described in detail in Section 5.3.1, which, according to the participants, had a rather mixed effect on the results. However, since the output as a whole was not that relevant, this information does not necessarily mean that the functionality itself was not working properly. Instead, it could be that the slider was performing its function as expected but because of the poor results produced by the underlying clustering model, this was not as apparent to the end users. The study subjects also had a mixed opinion about which part of the output was most helpful - the suggested solutions or the similar issues. Of course, given the fact that they were often unable to find what they were looking for at all, this sort of uncertainty is completely
understandable.

All in all, even though the results of the user evaluation were rather negative, they were nevertheless extremely valuable, because they helped identify the most glaring flaws of both the design and the implementation of the proposed solution, thus essentially providing a comprehensive blueprint of how it can and should be improved in the future. However, the fact that the dataset used for the training was quite small, especially compared to the actual number of issues which could be found in all the JavaScript repositories on GitHub, should be taken into account when analyzing the outcome of the evaluation sessions. For an even more in-depth analysis of the evaluation outcome, see Section 7.2.3. On the other hand, for a thorough presentation of the results obtained during the user study, please refer to Appendix F.
7 Discussion

This chapter presents some reflections regarding the developed solution as well as the results obtained from the various evaluation studies. It discusses both the shortcomings and the strengths of the implementation and explains how the system can be improved in the future. What is more, the chapter also provides an interpretation of the evaluation results and tries to address the different talking points raised by them. Finally, the last section outlines some of the limitations that, to some degree, affected the overall outcome of the research.

7.1 Implementation

As mentioned on several occasions, the specific implementation that was developed as part of the work presented in this thesis, was not the main focus of the study, given the fact that the research had an exploratory nature and aimed to address a particular problem, namely facilitating the automated analysis of issues based on their user submitted description. Therefore, the exact tools that were used in order to accomplish this goal were not of utmost importance. Nevertheless, it is still crucial to reflect upon the various choices that were made throughout the study process as well as objectively assess the strengths and weaknesses of the current implementation in order to better understand how it can be improved in the future. This information will be beneficial not only for anyone who chooses to build upon the efforts presented in this thesis, but also for other research studies that may decide to tackle the same subject.

7.1.1 Data

The most significant shortcoming of the developed solution was the fact that the data, which the system was working with, was rather limited - the training dataset only contained issues from a relatively small number of JavaScript repositories, especially taking into account the total amount of data that is available on GitHub as a whole. Even though the results of the various evaluation measures that were taken in order to assess the main aspects of the implementation, such as its capacity to separate issues into self-contained categories and then accurately assign new entries to the appropriate group, were very promising, they could still have benefited from a larger training dataset. For instance, the analysis of the clustering results, by both the author as well as the two independent groups of external reviewers, showed that only around half of the clusters created during the model training had a clearly distinguishable main theme. Although this outcome provides a positive indication of the feasibility of the proposed solution, if the training procedure was done with a more sizeable number of issues, that would allow the algorithm to discover even more subtle similarities or differences among the various entries and thus perform a more accurate clustering of the items, as well as offer a group of categories that could be generalized to a large number of
7. Discussion

JavaScript repositories.

Despite the fact that the limited amount of data used for the model training had a considerable impact on every aspect of the implementation, its effect was most palpable on the web prototype tested during the user evaluation stage of the study. Due to the fact that all of the data that was being displayed could only come from a small pool of repos, many search queries simply could not produce any relevant output. For example, searching for an issue about a project that has not been considered during the initial data collection stage, is an endeavor doomed to fail. It could be argued that, since these shortcomings were known prior to the user evaluation, the procedure should have been designed in such a way so that such factors do not affect the study results. For instance, there could have been stricter rules for the participants regarding what types of issues they could use, e.g. only issues about projects which are known to be in the training dataset and/or concerning a specific problem instead of a general inquiry. However, imposing all of these restrictions would have effectively defeated the purpose of involving users in the evaluation process at all, without whom, it would not have been possible to objectively assess the real world applicability of the prototype. Nevertheless, one alternative approach might be to do as one of the participants involved in the study suggested and rephrase the queries so that they are formulated more like actual issues, although it is still unclear whether this would have a noticeable impact on the results.

7.1.2 Design approach

Another problem of the system concerned some aspects of the chosen design approach. For instance, a lot of the top rated comments written in response to various issues, did not actually represent a solution to the posed problem but were instead statements indicating that the issue will be or has already been addressed. Therefore, if a given issue occurs often or has a significant impact, such a response would get a very positive reception from the community, gather a high number of upvotes and appear as a search result in the prototype implementation. However, since this type of a post does not contain any pertinent information regarding how the issue it relates to can be solved, it does not provide much value for the users of the application. Moreover, one of the most common use cases for the “Issue” section of a GitHub repository is for people to report different problems that they have experienced using a specific library or framework, directly to the developers who are working on the project. Furthermore, sometimes it turns out that these problems were not caused by the fact that users were not utilizing the library as intended but instead by actual bugs in the code. As a result, many of these inquiries are addressed by applying a fix to the source code, which presumably solves the issue. Nevertheless, since the exact solution does not really concern the end user, it is usually not explained at length in the comments. This means that a large portion of the issues that are being posted, do not include actual solutions in the comments associated with them. Therefore, with all of this
in mind, it becomes apparent that the approach of extracting possible solutions from the responses would be better suited for a Q&A platform such as StackOverflow, where there is a distinction between answers and comments and so it would be possible to retrieve only the posts which do in fact address the problem at hand. Furthermore, the site is heavily moderated in order to ensure that both the questions and the answers are of sufficient quality and somehow contribute to the general knowledge pool. One possible solution to this problem would be to utilize a similar strategy to the one employed by Badashian, Hindle and Stroulia (2016), and use the terms in the title of the submitted issue so as to find relevant answers directly from StackOverflow. Due to the aforementioned benefits of the platform and taking into account the feedback received from the user evaluation, which indicated that people use SO as a primary source of information related to programming issues, the likelihood of finding relevant answers to user queries would almost certainly be much higher.

With all of this in mind, it can be concluded that this sort of an implementation might not be the best example how natural language processing and machine learning can directly benefit the users of the GitHub platform, even if every component has been perfected and produces the most optimal output. Nevertheless, even though the chosen approach might have been flawed, that does not necessarily mean that it did not provide any valuable information. Despite the fact that the search results might not have contained a direct answer to the users’ query, they still could, at the very least, point them in the right direction and aid them in their search for a solution. For example, displaying a link to the full issue thread of a given comment, allows users to not only immediately recognize whether the answer is relevant for them (by seeing which repository it comes from), but also to browse through all responses and see if any of them offer any meaningful contribution to their own case. Moreover, even though, as mentioned, the most upvoted answers often represented a statement that did not directly address the issue itself, many of them still contained a link to a different entry which might be relevant. Last but not least, the implementation itself can always be revised and enhanced, for instance, by incorporating semantic analysis of the comments, which would help to more accurately determine their content and thus their relevance to the search query.

The other main function of the web application - finding the most similar issues based on the user query, can be viewed as analogous to that of a regular search engine. Therefore, why would anyone use this application instead of a search engine with a proven track record which is also more likely to provide a relevant result? One of the main strengths of a search engine, such as Google, is the fact that it works with an extremely vast amount of data. Even though, the exact details of exactly how Google operates are not known to the general public, it is safe to assume that it incorporates some kind of a combination of natural language processing and machine learning in order to process a search query and produce the best possible response. Therefore, having such an enormous quantity of
data at its disposal, it can be utilized to train the search engine and thus ensure that it can better recognize the intricacies of human speech in order to provide the most relevant results. However, the primary strength of these general-purpose search engines is also their biggest weakness. They have indexed billions of documents covering a multitude of different subjects and domains, which means that unless a query is very specific, which is rarely the case when it comes to searches, the system must consider a wide range of possibilities and try to determine which of them is the most relevant. Naturally, Google is incredibly good at that, even offering a service called “Google Enterprise Search”[^30], specially designed for businesses so that they can utilize the power of the Google platform to search solely through their own data. However, while the search engine is only indexing the data for search purposes, it likely does not perform any categorization of the results (or the query), which, as evidenced by the outcome of the expert analysis sessions, is one of the main strengths of the developed solution. Furthermore, a system that is trained with only a particular type of data will perform even better when trying to find a relevant answer, because there would be little to no “noise” that could affect the search results. Besides, since all the data that is being analyzed comes from a specific domain, many terms would not have the ambiguous meaning that they would otherwise, if they are taken in a more general sense. For instance, words like “library”, “package” or “model” would have a much more narrow usage and thus it can be immediately concluded exactly what they refer to, even without considering the context in which they are being used.

Unfortunately, as explained earlier, at its current state, the proposed system does not have the “luxury” of working with such a large sample of data, which is why, even though the results that are being returned show some promise, there is still a lot of room for improvement. However, given that the overall design is sound, the workflow logic is correct and each component is working as it should, the machine learning model is bound to become even better and even more accurate, as more data is introduced to it.

### 7.2 Results

As discussed in [Chapter 6](#), there were a number of evaluation procedures which were conducted in order to thoroughly evaluate the developed solution. These assessments provided an abundance of useful information about different aspects of the system and revealed that even at such an early stage in its development, it can still produce valuable output that helps address the main research questions posed at the beginning of the study. This section reflects upon the results obtained from each of these evaluations in more detail.

7. Discussion

7.2.1 Expert analysis

The expert analysis sessions were arguably the most exhaustive of the evaluation studies performed as part of this research. Therefore, the results attained from these sessions can also be viewed as the most reliable, due to the involvement of two separate groups of external reviewers, both from the academic and the business sphere. As discussed in Section 6.1.4, their label assignments illustrated that there was an overwhelming consensus among them regarding the fact that a significant portion of the resulting clusters do indeed represent distinct issue categories. This insight serves as evidence that the system can help discover patterns within the data, which is in itself, a major contribution that could be utilized in a number of ways, such as issue triaging, automatic labeling or pre-emptive issue resolution, as suggested by the participants in the expert analysis. Furthermore, as mentioned in Section 1.1, such a solution can provide developers with a way to oversee their projects and quickly identify potential problem areas. For example, the owners of a given GitHub repository could use the developed solution to analyze their repo and find out that a large portion of the submitted issues were about the documentation of the project, thus signifying that this could be something that needs to be addressed. As a result, they could decide to focus more of their efforts on improving their documentation so that they can provide a better experience to their user base. Of course, the proposed solution is not solely limited to web based applications and can be utilized in other contexts as well. Apart from the possibilities mentioned by the experts during the analysis sessions, another example would be the programming related email threads that people can subscribe to in order to receive updates about different questions and answers that are being posted regarding a specific topic. Using the developed system, it would be possible to automatically analyze a new thread submission and immediately send an email to the poster containing possible solutions to his query. In this scenario, the entire process would be carried out without any human involvement, thus not only speeding up the proceedings but also, arguably providing users with better, more relevant answers (once the system has been perfected). Lastly, another example scenario for future use could also be in exploring the behavior of chatbots, so that they can better address the requests and inquiries of the people interacting with them.

Nevertheless, since the experts were presented with only a subset of the total number of clusters created during the training process, the results of the analysis do not provide definitive proof of the model’s overall accuracy, especially given the fact that they were only introduced to the clusters which were most well-formed. However, even though the reviewers did not examine all the data in its entirety, their analysis still serves to validate the feasibility of the chosen development approach, which due to the exploratory nature of the study, is a crucial insight. Furthermore, as mentioned in Section 6.1.4, the experts were only given a portion of the clustering data in order to reduce their workload, meaning that some clusters which were also deemed to be well-formed were excluded as well, despite the fact that they could have provided an even stronger case for demonstrating the accuracy
of the clustering model. The analysis also showed that some clusters are not defined well enough and the terms contained within them appear to be too mixed. Therefore, the clustering process would either have to be conducted with a larger amount of input data as mentioned in Section 7.1.1 or the clusters which are not homogeneous enough would have to be broken down further. However, even though the categorization process is complex and time/resource consuming, once the model has been trained, it would only have to be periodically adjusted, as suggested by Jonsson et al. (2015), who found that outdated issues can bias the accuracy of the learning model. So in order to remedy this, they propose for old issues to be regularly discarded at a given interval of time and replaced by new ones which are more relevant to the current state of the fast-paced software industry.

Finally, despite the fact that the clustering model did not manage to produce as accurate results as some of the implementations developed during the research studies mentioned in Chapter 2, it should be taken into account that by utilizing an unsupervised learning approach (unlike the aforementioned systems), the proposed solution fully automates the categorization process, which is one of the most significant contributions of the whole thesis. This means that the solution virtually eliminates the need for manually analyzing large sets of data in order to classify them, which naturally comes at the cost of reduced accuracy and can thus be seen as a certain trade-off. Nevertheless, given the fact that the implementation is still in an early development phase, it can be considerably improved in the future so as to alleviate some of its potential shortcomings.

7.2.2 Issue assignment

One of the main insights to come out of the issue assignment evaluation procedure was that some of the issues which were “misclassified”, were put into clusters that were closely related to the expected assignment, but were not formally labeled by the experts. Furthermore, most of these same clusters were also deemed to be well-formed during the initial cluster analysis. This means that, had they also been examined by the external reviewers and given a definitive label, they could have been considered during the labeling of the evaluation data. As a result, some of the issues which were misassigned, could have been classified correctly. Moreover, as it was the intention from the start, clusters which are very similar to one another, could be merged together under a single label, thus improving the accuracy of the classification even further. Last but not least, as evidenced by the work of Chawla and Singh (2015), the categorization could be done with groups that are not necessarily mutually exclusive, meaning that any given issue can belong to more than one category at the same time. This approach offers a much better representation of real world data, which is often ambiguous and difficult to define using a single term. For example, one of the analyzed issues which had the following title - “Update documentation to change install instructions to use npm”, can correctly be given three of the labels that were defined as a result of the expert analysis phase, namely “Documentation”, “npm” and “Package
installation”, since the issue seems to be about all of these subjects. However, using the current clustering implementation, an issue can receive only a single label, so it is up to the discretion of the reviewer to decide which of the aforementioned labels is the most suitable in this particular case. Moreover, if the model assigns a different label than the one given by the human expert, the classification would be considered as incorrect even though it was not. Nevertheless, the possible implications of such a mistake in a practical scenario would have to be investigated further.

What is more, despite the fact that a number of measures were taken in order to eliminate bias and ensure that the obtained results are reliable, since the evaluation was done by the author, its findings would need to be confirmed through an additional, more rigorous study, conducted in a manner similar to the expert analysis sessions, with the help of external reviewers who can more objectively label the evaluation data. Having said that, the outcome of this procedure still serves as an initial indication regarding the assignment accuracy of the trained model. Furthermore, since it also corresponds with the outcome of the expert analysis sessions, together the results of these assessments demonstrate that the proposed solution can effectively categorize issues based on their textual description.

7.2.3 User evaluation

As shown by the results of the user evaluation study, people are mostly used to finding solutions to their programming issues through the use of search engines and Q&A’s, which have good SEO and have been designed so as to facilitate the process of finding answers to specific queries. In fact, since virtually every online search begins through a search engine and a site like StackOverflow invariably appears at the top of the results, the two platforms are inseparable. Compared to them, GitHub is not as well-suited for providing answers to programming related queries, not necessarily because they cannot be found on the site, but because of a number of other factors. For instance, the visual layout through which issues are presented on GitHub varies significantly from the way similar entries would appear on StackOverflow. On GitHub, the issues and the entire discussion that follows them are shown like a forum thread - the original entry is at the top, while the replies are positioned below and are a little indented so as to indicate their relation. Furthermore, the entries are sorted chronologically, with the oldest posts appearing at the top and the newest ones at the bottom. On the other hand, StackOverflow does not rely on such a rigid structure - answers are usually sorted by their rating, with the most upvoted posts showing first, followed by all the rest. However, if the original poster has marked as “accepted” an answer which is not the most upvoted, then that becomes the response which is shown at the top. This layout makes it extremely easy for visitors to quickly locate the information they are looking for and move on with their work. Unfortunately, the structure utilized by GitHub - the fact that the best rated comments do not stand out in any way, the ordering of the responses, etc., does not facilitate the browsing process, which makes finding relevant information more
7. Discussion

difficult for visitors. This, combined with the seemingly worse SEO that the site appears to have, has driven people away from the platform. Furthermore, this general wariness of GitHub as a solution provider might also have an effect on the way people perceive it, which could also have, to a certain degree at least, influenced the assessment of the user evaluation participants regarding the proposed solution. Of course, as shown by the aforementioned arguments, their reservations are not without merit - currently GitHub is not on the same level as StackOverflow, when it comes to providing solutions, nor does it aim to be. The “Issues” section in each project hosted on the platform, for the most part, serves a wholly different purpose - to give users the ability to provide the developers with direct feedback about their product and potentially even get involved in the creative process, either by contributing with ideas for possible improvements or even source code. Nevertheless, the potential pre-existing bias that users might have had toward GitHub should be taken into account when analyzing the results of the study. Moreover, since each GitHub project has its own dedicated “Issues” section, different repositories have chosen to utilize it differently and have imposed their own rules regarding the types of entries which are allowed to be submitted. As a result of this decentralized system, both the quality and the content of the posts, can vary widely on a project-to-project basis. This means that the issues that exist on the platform can be very diverse and so one cannot easily make any definitive conclusions regarding what could be found there and what could not.

Furthermore, even though the evaluation showed that the prototype was not well-received by the users, since it did not help them accomplish their goals, it also demonstrated some of the positive aspects of the implementation, which indicate that overall, the solution is headed in the right direction and even though it is not currently ready to meet these specific needs, it can very well do so in the future. What is more, the fact that end users are, for the most part, entirely focused on the immediate benefits that a given application can provide them, means that they rate the usefulness of that application using a much steeper grading curve and are in general a lot more critical, unlike researchers and practitioners who can better understand that the product is still a work in progress and are more capable of appreciating its potential rather than being completely fixated on its current flaws. However, the specific reason for this disparity in the assessment between the expert analysis and the user evaluation sessions, could be the fact that finding issue solutions is a much more complex and fine-grained task than issue categorization. Moreover, as mentioned in Section 7.1.1, the data that was used for the training was rather limited, which served to significantly restrict the variety of the results which were shown to the users. Therefore, it may still be possible for the chosen approach to be utilized in order to accomplish these tasks, but it is just not at this stage yet and needs further improvement before it can be used for both. Nevertheless, at the current time, there is simply not enough evidence to make a definite conclusion, so further research efforts would have to be undertaken so as to address this particular question. One thing that is certain though is that, at its present
state, the web prototype has a limited external validity and requires considerable enhancement before users would have enough of an incentive to adopt it instead of other more established alternatives.

The feedback from the user evaluation also confirmed the assumption that, when it comes to programming related solutions, people prefer code instead of in-depth explanations or links to other resources. The reason for this is that in today’s day and age, people have become accustomed to instant gratification so much so that they would often choose an option that is quicker even though it might not necessarily be better. The appeal of a simple (or even not so simple) code snippet is that one can quickly copy it, paste it directly into a text editor and immediately observe its effects. It takes little to no time to use and requires essentially zero grasp of the nature of the problem and why this particular solution may or may not solve it. Of course, there are still many excellent code snippets that could be found, especially those which have accumulated a large upvote count. In these scenarios, the positive votes serve as a sort of “seal of approval” from the community which signifies that a given entry is both relevant and useful. Therefore, a future version of the application can make use of the fact that people have a general preference for code samples as opposed to long, detailed explanations, and consider the presence of code within the suggested answers when deciding which entries should be given more “weight” and thus be presented at the top of the results. For example, if a given comment contains a code snippet, it would be given preferential treatment compared to others that do not because, from the perspective of the end users, such answers are deemed to be more useful.

7.3 Limitations

The main limitation which dictated a number of decisions made throughout the study period was the insufficient memory and processing capabilities of the machine used for developing the solution. On a number of occasions, these hardware limitations had a major influence on the final design choice that was made, which was not necessarily the one that was considered to be the most appropriate in the given situation. Therefore, the overall workflow and implementation can easily be improved by merely using a more powerful machine, which can better perform the various tasks required for the development of the solution. Another alternative could also be to make use of a machine learning library like Apache Spark\textsuperscript{31}, which would help solve these problems because of the fact that it utilizes a distributed computing approach which drastically reduces the time and processing power required to handle large datasets. However, a decision was made not to adopt the library as part of this research, due to its steep learning curve. Furthermore, the machine learning components offered by the library are not as feature rich or as customizable as their counterparts in scikit-learn.

\textsuperscript{31}http://spark.apache.org/
Another limitation came from the author’s lack of expertise in the field as well as the rather basic familiarity with the different technologies utilized in the implementation. What is more, the subject matter itself is extremely complex, primarily due to the fact that the problem that is being examined is also far from trivial. Besides, at every point of the development process, there were many aspects that needed to be considered and many possible options for doing even a rather simple task, which was at times very overwhelming. All of these factors had a certain effect on the research process and the final solution, which as evidenced by the various evaluation sessions, can be improved in a number of ways.
8 Conclusion

This thesis presented a detailed account of the research conducted so as to explore how to analyze open source project issues based on their textual description, by utilizing a combination of natural language processing and machine learning techniques. In order to address this particular topic, an initial proof-of-concept implementation was developed which made use of the aforementioned methods for examining issues collected from a wide range of open source JavaScript repositories hosted on GitHub. Through this analysis, the system aimed to identify and group together distinct types of issues and possibly find suitable solutions to these problems. The final solution was evaluated with the participation of both experts in the software field as well as users representative of the people who can primarily benefit from such a system. These assessments served to validate the chosen approach for tackling the problem of automated issue analysis and indicated that the implementation can effectively separate issues into self-contained categories based on their type. This chapter first explains how the outcome of the study helps address the research questions posed at the start of the thesis work. Furthermore, it provides a summary of the specific research contributions made by this study and outlines some potential directions for improving the developed solution as part of future scientific efforts.

8.1 Research questions

The research questions first introduced in Section 1.3 exemplified the specific goals that this thesis aimed to achieve. The main study question which guided the work efforts throughout the development process, was formulated as such:

**RQ:** How can natural language processing and machine learning techniques help analyze issues gathered from open sources project repositories hosted on GitHub based on the user generated description of each issue?

Due to the fact that this question was rather broad, it was divided into three more specific inquires, as follows:

**RQ1:** How can these techniques facilitate the process of identifying distinct issue categories?

Since answering this question could lay the foundations for addressing all the rest, it was undoubtedly the main focus of the study. Therefore, in order to provide an answer to this inquiry, a rather sophisticated system was developed, which thoroughly cleaned and prepared the data extracted from GitHub using a variety of natural language processing techniques, before submitting it to a clustering machine learning model, which grouped issues based
on their similarity. Then, the resulting clusters were evaluated by two independent groups of external reviewers who predominantly agreed that these groupings do in fact represent distinct issue categories, thus answering RQ1.

**RQ2:** Which natural language processing and machine learning techniques can be applied to automatically assign an issue to a particular category?

Since RQ2 was closely related to RQ1, the assumption was that the same strategy used to address RQ1 can be utilized again. First, a new set of issues was collected, labeled, transformed through the same NLP methods as before and submitted to the clustering algorithm trained for the initial categorization. Then, the issue labels given before the analysis were compared with the ones produced by the trained model, which showed that the model can, for the most part, reliably assign issues to the categories in which they belong, thus addressing RQ2.

**RQ3:** Which computable approaches can be utilized to identify possible solutions that are relevant for the identified issue categories?

Finally, in order to determine if the proposed solution can be used for identifying relevant answers to programming issues, a dedicated web prototype was developed, again largely based on the clustering model that was previously trained. The prototype first assigned a new incoming issue to a given category and then attempted to find a useful solution within the comments that were posted in response to the most similar issues found within said category. Furthermore, in order to facilitate the search for possible solutions, a hierarchical subcluster tree structure was added to the model, which allowed users to specify at which level of the clustering hierarchy, the solution should be found. However, an evaluation involving several representative users showed that, even though, the prototype was sometimes able to provide relevant output, it could not perform its primary function to a satisfactory level. Nevertheless, due to a number of factors, such as the limited amount of data that was used for the model training, this outcome still does not provide a clear answer to RQ3.

Overall, the insights gained from the work done so as to address each individual research question helped provide an answer to the main research question (RQ) of the study. The findings of this research demonstrate that natural language processing and machine learning techniques can indeed facilitate the automated analysis of open source project issues based on their user submitted description and this thesis describes a detailed workflow through which this could be accomplished.
8.2 Contributions

There are a number of research contributions that this study provides, as first outlined in Section 1.5. First and foremost, this thesis fills a certain knowledge gap that exists in the area of categorizing issues collected from GitHub based on their textual summary. Furthermore, this is accomplished by utilizing an unsupervised machine learning algorithm i.e. clustering, without relying on any pre-labeled data in order to form the issue categories, thus significantly facilitating the entire process, since such data is extremely hard to attain in a real world situation. What is more, the research serves to demonstrate that combining natural language processing and machine learning methods is a viable approach for analyzing unstructured data, which can help find distinct patterns within it and thus identify common programming problems. This insight can in turn provide a more in-depth understanding on how to maintain and improve existing as well as future open source projects, which can benefit the software development community as a whole. Moreover, since these techniques have not been studied before, with regards to their suitability for examining issues posted on the GitHub platform, the research provides a novel contribution in this particular area as well. Last but not least, the work performed as part of this study resulted in the creation of a functional implementation which incorporates the aforementioned NLP and ML techniques and demonstrates how they can be effectively utilized in a real world scenario.

8.3 Future work

Due to the fact that both the subject area and the topic of this thesis is so vast and complex, the work being at display here could easily be developed into a full PhD thesis. Moreover, since the purpose of the developed solution was to serve as a proof-of-concept prototype, there are many aspects of it which can be built upon. Therefore, this section presents a list of areas in which future efforts could be invested to further improve the performance of the system and its application in various real world scenarios.

One possible enhancement would be to perform a more thorough preprocessing of the data prior to the clustering phase, for instance, by incorporating a more extensive domain specific list of stopwords that includes, JavaScript syntax keywords. This will ensure that common words that do not carry much meaning in such a narrow context, will be discarded and will therefore not influence the resulting clusters. Such a measure will be especially useful if any future analysis also includes the body of the issues and not just the title. On the other hand, the preprocessing stage can also benefit from using another keyword list, but this time with the opposite purpose - instead of removing the words contained within it, giving them a higher weight, thus signifying their greater importance during the feature extraction and the subsequent clustering. A good candidate for such a list would be the names and aliases of different JavaScript libraries or frameworks. This will make sure that
the machine learning algorithm “understands” that they are of higher significance compared to other terms, which will be reflected in the eventual clustering separation.

Another improvement to the clustering procedure would be to utilize semantic analysis of words, when determining whether two documents are similar to one another, which would enable the detection of synonyms, idiomatic expressions and so on. As Manning, Raghavan and Schütze (2009, p.413) point out, dealing with synonyms as well as words which have multiple meanings depending on the context, is an inherent limitation of the vector space model. One example solution which makes use of such a technique is the word2vec\footnote{https://code.google.com/archive/p/word2vec/} algorithm. Moreover, this approach could help solve the problem mentioned in Section 5.1.4 where issues that address a very specific topic could not be categorized. The problem here is caused by the terminology that is being used to describe the issue (a problem also discussed with the experts during the analysis sessions) and therefore, by applying different techniques which can perform semantic analysis of the text in order to uncover its meaning, the issue can be grouped with others that have a similar connotation, despite the fact that they do not have even a single word in common. Nevertheless, it should be mentioned that there already is a small semantic component implemented as part of the prototype, namely the lemmatization facility which performs a morphological analysis of each word in order to transform it into its root form. Furthermore, as mentioned in Section 5.1.5, one of the chief benefits of the dimensionality reduction approach that was utilized, is that it can help alleviate the aforementioned problem to some extent. However, the specific effects of this measure, in regards to this particular matter, remain unclear and would most likely require a more in-depth investigation.

On the other hand, when it comes to the labeling of the resulting clusters, one option would be to employ a similar strategy to the one used by Carmel, Roitman and Zwerdling (2009), namely using an external source, such as Wikipedia\footnote{https://en.wikipedia.org/}, for automatically assigning an appropriate label to a cluster. This could eliminate the need for using a predefined taxonomy, which might be too general and difficult to adjust to a more specific domain, but would instead enable building the taxonomy categories based on the data at hand. Furthermore, this may allow a higher level of flexibility and, if the proposed labels are accurate and descriptive enough, could even serve to fully automate the process of cluster labeling, thus not requiring any manual analysis. This would have an added benefit because, as Eikvil, Jenssen and Holden (2015) point out, manual label assignments are often subjective and can vary significantly from person to person. Alternatively, the labeling process itself can be further facilitated by visualizing the clustering data so that the reviewers can more easily make sense of it. For example, Jusufi et al. (2014), utilized tag clouds as part of their visualization prototype in order to illustrate the most commonly used words within the text documents that were clustered together. This same technique could be applied so...
as to present the differential and internal terms extracted from each cluster and thus help address some of the problems that the experts experienced during their analysis.

Another possibility regarding the cluster labeling would be to adopt a combination of top-down and bottom-up approaches. At the top cluster level, a group of experts could still manually examine the different categories and try to assign a more general, descriptive label (i.e. the top-down approach). Naturally, at each sublevel, the number of clusters that need to be labeled increases exponentially, meaning that defining a label for each new category would require an excessive amount of manual labor. However, this problem could be solved by harnessing the efforts and collective intelligence of the “crowd”, that is, allowing users to specify their own categories for the various issues. This process is known as “collaborative tagging” but is sometimes also referred to as a “folksonomy”, a term which emphasizes the fact that the classification lies in the hands of the end users. As the former term suggests, the categorization is commonly achieved through the use of tags which are applied, in this case, to issues (directly) and clusters (indirectly). Of course, this approach also has some drawbacks such as the fact that for a certain period of time the system might not have enough collaborators to label the clusters, or it may suffer from the effect of incorrect tags being assigned by ill-intentioned users. One positive outcome of the bottom-up collaborative tagging strategy is that the collective intelligence of the users can also be harnessed to evaluate whether the top level labels assigned by the experts accurately describe, be it in a very general manner, the actual issues that the users encounter. For example, when a new search is made and the issue is assigned to a cluster, the top level category label, along with any other of the community generated lower level labels could be shown, and the users who submitted the query, could indicate if the classifications that are offered are correct or not, by accepting or rejecting the suggestions and possibly even providing their own categorization.

Finally, an alternative development strategy which might be worth investigating in a future study, would be to adopt the approach utilized by Pappuswamy et al. (2005), Kyriakopoulos and Kalamboukis (2006) and Lin and Wu (2009), first mentioned in Section 2.4. As explained there, they used clustering as a preprocessing step in order to group similar data items and thus improve the accuracy of the trained classification model, which as the literature showed, is quite a common strategy. Since the work performed as part of this thesis has already provided the infrastructure, by separating text documents into clusters and labeling the resulting categories, one possible direction for future research could be to examine if adopting an additional classification step can help to further improve the accuracy of the issue assignment. Another paper that adopted a similar strategy was the study of Pingclasai, Hata and Matsumoto (2013), who chose to make use of a categorization technique called “topic modeling” for supporting the process of text classification. As the name suggests, this method tries to extract a set of “topics” i.e. a small group of words that frequently occur together, from a collection of text documents and thus potentially
discover a semantic relation between them. Therefore, investigating whether topic modeling, or any other technique not discussed in this thesis, can be a viable alternative for identifying distinct issue categories is another interesting subject for future research.
References


References


Dhillon, I. S., Guan, Y. and Kogan, J. (2002). Refining clusters in high-dimensional text data. In *Proceedings of the Workshop on Clustering High Dimensional Data and its Appli-


GitHub Developer. n.d. Reaction types. [image online] Available at: https://developer.github.com/v3/reactions/#reaction-types [Accessed 01 June 2017]


References


Appendix A Literature review protocol

This section describes the protocol that was followed during the literature review stage of the research, which aimed to discover more information regarding the use of natural language processing and machine learning for automated analysis of issues and bugs. As mentioned in Section 3.2, the review adhered to the guidelines suggested by Kitchenham and Charters (2007).

A.1 Research questions

The main purpose of the research questions (LRQx) that the literature review aims to address is to learn more about the most integral topics covered in the thesis and provide a more in-depth understanding of the subject matter, so that the gathered knowledge will serve as the basis for the subsequent research efforts. Moreover, the information acquired during the review would help guide the most crucial decisions and design choices throughout the whole study and thus help answer the main research questions of the thesis, outlined in Section 1.3. That being said, here are how the literature review questions are formulated:

LRQ1: What is the state-of-the-art within the fields of natural language processing and machine learning?

Since this research question is rather general, it can be further divided into several questions which address a more specific aspect of the subject, namely:

LRQ1.1: What are the usual applications of these methods?

Through this first question, the review aims to establish how natural language processing and machine learning have been utilized within the scientific community and for what purposes. This will help to better understand the benefits that these techniques are able to provide and how they can be applied in a real world context.

LRQ1.2: In which scientific fields are they most commonly being used?

The second research question is a natural continuation of the first. Once it is determined how these methods have been utilized, the next logical step is to examine which scientific disciplines make use of them on the most regular basis as well as to investigate what are the reasons for this phenomenon. This information could point to some aspects of these techniques that might make them more suitable for one subject area compared to another and thus provide an initial indication about whether they could be effectively applied in this research.
LRQ1.3: What are the specific techniques and algorithms that are being utilized?

Finally, the last sub-question focuses on the specific methods and even algorithms that have been used in these studies. This will help figure out which particular techniques are appropriate for accomplishing a given task and which might not be - information that could be vital during the design phase of this research.

While the aforementioned research questions aim to provide a more general overview of the main study subjects, the rest focus on much more specific aspects of the topic so as to dig deeper into the matter and thus help acquire a more in-depth understanding of it all. Having said that, here are the rest of the literature review research questions:

LRQ2: Have natural language processing and machine learning techniques been applied for analyzing bugs and issues?

This question intends to investigate whether such methods have been used before in order to achieve a similar goal, namely analyze software related bugs and issues. If that is the case, these previous studies can provide a guideline on how to address this particular topic as well as outline the potential downfalls that should be avoided.

LRQ3: Are there any issue taxonomies that have been established within the software development field?

Another interesting aspect that the literature review aims to address is the possible existence of any established and well-accepted issue taxonomies within the software field. If there are such classifications and they appear to be suitable for the task at hand, they could perhaps be utilized for the categorization of the issues.

LRQ4: Have there been any similar implementations to the one that is to be developed?

Last but not least, the final research question focuses on discovering whether there are any similar studies that have been done in the past and whether any comparable solutions have been previously developed. This will help estimate the novelty of the current research and understand how it fits within the scientific field as a whole.

A.2 Search scope

Since the goal of the literature review is to establish the state-of-the-art in a variety of subject areas, the only papers, books or other scientific publications that are being considered are those published from the turn of the millennium (the year 2000) up until the
end of 2016, due to the fact that the work on this thesis begins in early 2017. Therefore, any paper which has been published before or after the specified time frame, would not be taken into account, meaning that some relevant publications could potentially be overlooked, especially the most recent ones (published after the date of this study), which are more likely to provide a novel contribution to the field due to their recency. Nevertheless, the aforementioned time constraints are generous enough so that the majority of relevant research would be represented. Besides, any missing papers could potentially be examined as part of some future research efforts.

Furthermore, other publications which are not being considered include editorials, abstracts and short papers (less than at least 5 pages), due to the fact that they do not cover the given topic in sufficient depth. What is more, publications which are not particularly related to the research questions or which do not provide a meaningful contribution, would also be discarded. In summation, all papers that do not fit the criteria outlined below, will be excluded:

**Inclusion criteria**

- Published between 2000 and 2016
- At least 5 pages long
- Relevant to the research questions and/or providing a meaningful contribution

However, if some paper that does not meet these requirements appears to be extremely influential and is often cited and referenced in the resources that are found, it is possible to make an exception and include it as well. Those additional papers will be explicitly referred for review and replication purposes (see Appendix B).

**A.3 Data items**

“Data items” is a term signifying the specific pieces of information which are collected as a result of a literature review. In this case, this information will serve a number of purposes such as answering the research questions outlined in Section A.1 guiding the decision-making process throughout the entirety of the study and so on. Table A.1 presents all data items that are to be gathered from analyzing the various publications found during the review, along with the corresponding rationale for their collection.
Table A.1: A summary of the different data items gathered as a result of the literature review, along with the specific reasons for their collection

<table>
<thead>
<tr>
<th>Item</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Bibliography</td>
</tr>
<tr>
<td>Title</td>
<td>Bibliography</td>
</tr>
<tr>
<td>Year</td>
<td>Bibliography, search scope</td>
</tr>
<tr>
<td>Length</td>
<td>Search scope</td>
</tr>
<tr>
<td>Developments in the NLP and ML fields</td>
<td>LRQ1</td>
</tr>
<tr>
<td>Application/usage of NLP and ML techniques</td>
<td>LRQ1.1, LRQ2</td>
</tr>
<tr>
<td>Scientific field/domain</td>
<td>LRQ1.2</td>
</tr>
<tr>
<td>Specific NLP/ML methods and techniques being utilized</td>
<td>LRQ1.3</td>
</tr>
<tr>
<td>NLP/ML methods used for issue/bug analysis</td>
<td>LRQ2</td>
</tr>
<tr>
<td>Issue taxonomy/classification/categorization</td>
<td>LRQ3</td>
</tr>
<tr>
<td>Software implementation</td>
<td>LRQ4</td>
</tr>
</tbody>
</table>

For example, information about the paper’s authors, title and year of publication is needed so that it can be properly referenced in the bibliography of the thesis, and in the case of the year and length of the paper - in order to ensure that it falls within the scope of the literature search described in Section A.2. Furthermore, as mentioned, other data items like the scientific field covered in the paper or the software implementation that was created as a result of the study, serve to address the research questions posed earlier, thus helping gain a more in-depth understanding of the subject matter and its various aspects and complexities.

Naturally, not every paper is going to contain information regarding all the data items shown in Table A.1 due to the fact that, despite being related to one another, these topics represent a wide range of research areas, with each being subject to extensive scientific study. However, gathering a variety of papers that address one (or more) of these topics in detail, helps cover all items, while at the same time allowing to delve deeper into each separate subject.

A.4 Search strategy

The search itself is conducted using primarily online resources, namely Google Scholar\textsuperscript{34} and the LNU digital library database\textsuperscript{35}, both of which grant access to a large number of scientific databases. What is more, both platforms allow for highly advanced and customizable search queries to be made, thus facilitating the process of finding the desired resources. However, only the first 5 pages of the search results are being considered since most of the papers

\textsuperscript{34}https://scholar.google.com/
\textsuperscript{35}https://lnu.se/en/library
which appear after that point are usually not particularly relevant to the original search query. Therefore, another, more fruitful strategy is adopted i.e. slightly adjusting the query itself (if needed) and then examining the first 5 pages of the new result set. What is more, for each of the search results, or as Kitchenham and Charters call them “primary studies” (2007), that are found to be relevant, the publications mentioned in their respective reference list should also be examined.

In order to limit the amount of search results and ensure that they are as pertinent as possible, the terms used in the search query have to be present either in the title or the abstract of the paper. Furthermore, each query is repeated several times using synonyms of the search terms, since many different phrases can often refer to the same concept or artifact, meaning that some relevant resources could be overlooked due to the fact that the query did not contain a specific word. For example, the term “natural language processing” is often used interchangeably with “text mining” or “information retrieval”, while “machine learning” is sometimes also referred to as “data mining”. Table A.2 lists some of the specific keywords that are to be used for addressing each of the research questions posed in Section A.1. It is important to note that a large portion of the words specified for each question can also be used for some of the others so, for instance, the keywords used for LRQ1 (and its sub-questions) can be used for LRQ2 and LRQ4, the keywords for LRQ2, for LRQ3 and LRQ4 and so on. Of course, the list could not be completely exhaustive since the literature review will continue throughout the research process in order to help address a variety of topics and challenges encountered during the study. What is more, the findings of the initial review activities will serve to guide the subsequent search queries and thus dig deeper into the matter at hand.

Table A.2: An overview of the various search keywords and phrases used to retrieve papers, which could help address the literature review research questions

<table>
<thead>
<tr>
<th>Research question</th>
<th>Search terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRQ1</td>
<td>natural language processing, text mining, information retrieval, text analytics, machine learning, data mining, knowledge discovery, application, methods, techniques, use, usage, algorithms, approach</td>
</tr>
<tr>
<td>LRQ2</td>
<td>software, bugs, issues, repositories, analysis, examination</td>
</tr>
<tr>
<td>LRQ3</td>
<td>taxonomy, categorization, classification</td>
</tr>
<tr>
<td>LRQ4</td>
<td>category identification, recognition, issue assignment, solution, resolution, answer, suggestions, GitHub</td>
</tr>
</tbody>
</table>
Appendix B Literature review results

During the second phase of the literature review, all papers that fulfilled the criteria outlined in Section A.2 were gathered along with publications which had been analyzed during previous research efforts, as long as they also met these requirements. Then, each publication underwent a rigorous evaluation process intended to establish whether it was relevant in relation to the research questions that the literature review was supposed to answer and if it met several quality criteria which aimed to examine the suitability of the research methodology adopted by the authors, whether they have clearly communicated how the study was conducted as well as the justification that they have provided for their design choices.

Every publication which has been used for the writing of this thesis is listed below. Section B.1 contains all papers that have been found as a result of the literature review, including articles discovered through references in the primary sources. On the other hand, Section B.2 consists of resources found through alternative means - be it through previous research or by being recommended by an external source.

B.1 Resources found through literature search


Appendix B. Literature review results


Appendix B. Literature review results


Appendix B. Literature review results


Appendix B. Literature review results

ACM SIGIR Forum, vol.31, no.SI, pp.74-81. ACM.


B.2 Additional relevant resources


Appendix B. Literature review results


GitHub Developer. n.d. Reaction types. [image online] Available at: https://developer.github.com/v3/reactions/#reaction-types [Accessed 01 June 2017]


Appendix C Expert analysis protocol

This section describes the protocol that was followed for conducting the analysis sessions with the external reviewers. It outlines the research questions that the evaluation aimed to answer and presents a detailed account of the exact procedure which guided these sessions.

C.1 Research questions

The main goal of the expert analysis is to validate the accuracy of the clustering model and establish not only whether it is able to produce clusters which have a clearly identifiable common theme, but also to what extent do the issues within it represent a specific type of problem. In other words, the study examines the potential utility of the clustering algorithm for performing semantic categorization of issues. The results of this analysis will help address RQ1 presented in Section 1.3, which is formulated as:

RQ1: How can natural language processing and machine learning techniques facilitate the process of identifying distinct issue categories?

Since these evaluation sessions are preceded by an initial cluster analysis described in Section 6.1.3, the results obtained through them can serve to either provide further support or disprove the findings of this preliminary examination. Naturally, due to the fact that, as mentioned in Section 6.1.4, the experts are only presented with a subset representing the most favorable part of the clustering data, the outcome of the analysis cannot be used to make definitive conclusions regarding the trained model, but will still serve as an indication of both its feasibility, i.e. the initial validity of the application design, and future potential. Therefore, the specific research questions that are to be addressed through the expert analysis (ERQx) are as follows:

ERQ1: Does the data found within each cluster have a clearly distinguishable overall theme?

ERQ2: Do the resulting clusters contain a distinct issue type?

The first question (ERQ1) attempts to determine not only whether the clusters are formed well enough so that the experts are able to agree on a particular label but also establish the level of confidence they have in the chosen assignment. Based on this, it can be concluded that the model was able to separate the issues according to a certain characteristic that they have in common, which provides a significant contribution towards answering RQ1. From this question, several hypotheses can be formed which describe the possible outcomes of the analysis in relation to ERQ1:
EH1.1: *The experts agree on the cluster labels and have a high confidence in their choice*

If the experts are able to agree on the particular labels and are certain about their choice, this would serve to demonstrate that the clusters do in fact have a clearly distinguishable theme which can be readily recognized. What is more, this will mean that the clusters are meaningfully representing distinct categories of issues.

EH1.2: *The experts agree on the labels, but are not very confident with their selection*

In case the experts reach a consensus regarding the labels, but do not feel overly confident about their selection, this would mean that the issues inside the clusters exhibit some commonality, but there is still some level of doubt when it comes to the overall categorization. This can be interpreted as the categorization having a predominant focus, but still requiring additional analysis, in order to narrow down the exact cluster semantics and/or divide them into smaller and more granularly defined categories.

EH1.3: *The experts disagree about the specific labels*

If the experts are unable to find common ground and agree on the labels which should be chosen for the resulting clusters, this would illustrate the fact that the clustering model needs further work before it can be used for issue categorization. However, if it is only a portion of the clusters that the experts cannot agree about, then only those categories would require any additional analysis. On the other hand, if a large majority of the clusters satisfy this hypothesis, this will reflect an issue with the overall classification and categorization design.

On the other hand, even though ERQ2 might appear to be very similar to ERQ1, there is a subtle difference between the two. While the first question aims to discover if there is a predominant pattern in the data that could be used to assign a descriptive label, the second tries to estimate if the data has high cohesion, meaning that all issues within the cluster are of the same type. If this happens to be the case that would provide even more evidence which could help address RQ1. Again, finding an answer to ERQ2 will be accomplished by relying on the feedback acquired during the expert analysis session. Based on the question itself, these are the possible hypotheses:

EH2.1: *The experts are very confident that the data is homogeneous*

If the reviewers all agree, with a high level of confidence, that the clusters contain a distinct issue type, this would serve to further validate the accuracy of the clustering model and sup-
port the claim that it is able to both separate diverse issues and group together similar ones.

**EH2.2: The experts are relatively confident that the data is cohesive**

If the experts are only somewhat certain that the issues inside the clusters are of a distinct type, this would indicate that even though the clustering model could divide issues which differ from each other, there is still some degree of “noise” within the clusters.

**EH2.3: The experts are not confident at all about the cohesiveness of the data**

Finally, if the researchers do not believe that the issues within the resulting clusters exhibit any sort of homogeneity, this would show that the clustering procedure might require considerable overhaul before it is equipped to effectively perform its function.

### C.2 Data selection

As mentioned on several occasions, the data used for the expert sessions is merely a subset of the data generated during the model training stage described in [Section 5.1.5](#). This was done due to the fact that during the preliminary pilot testing of the clustering results, it was determined that some of the clusters did not meet certain quality criteria and as such, it would be unnecessary for them to undergo further analysis. In fact, all of these clusters require additional work before it would be needed for them to be presented to an external reviewer. Nevertheless, the rest of the data, which is the focus of this study, can still be effectively utilized in order to determine the validity of the solution design.

The goal of the aforementioned quality criteria was to assess whether the different segments of the extracted data have something in common, which could help identify the main theme of the given cluster. There were three requirements which needed to be met in order for a cluster to be considered well-formed and thus be viable for subsequent examination. The requirements state that the following items have to share a similar theme among each other:

- The issue terms - acquired through the differential, internal and random methods (see [Section 6.1.2](#))
- The issue titles - internal and random
- The issue terms and the titles

If the data within a cluster met all of these criteria, then it was deemed to be suitable for additional review. In the end, out of the 30 clusters which were formed during the model training, 20 of them, which represent $\frac{2}{3}$ of the data, were found to be of sufficient quality based on the initial analysis. However, in order to reduce the workload on the
Appendix C. Expert analysis protocol

Before conducting any sort of evaluation, it is of utmost importance to develop a specific procedure that is to be followed throughout the entire time. What is more, having a predefined plan can also help avoid different types of bias which can have an adverse effect on the results of the entire evaluation. Therefore, this section describes all aspects of the study procedure prepared for the expert analysis sessions.

C.3.1 Study design

There are a number of measures taken while planning the procedure of the expert analysis, in order to reduce bias and thus improve the reliability of the findings. First of all, the protocol of the study has been reviewed by an external researcher, whose responsibility is to ensure that it follows scientific standards. Moreover, the evaluation is conducted with multiple people so that the effects of personal subjectivity are mitigated. Since the goal of the analysis is not to make any conclusive judgements, the exact number of participants is not crucial. However, a minimum number of study subjects will be required in order to have rich discussions, while at the same time aiming to determine the specific label of a given cluster as well as the confidence level of the experts. Furthermore, the people who are selected are all external reviewers who had not been involved in any capacity during the development of the project. As such, they would not be affected in any way by the outcome of the study, meaning that they could remain completely impartial. Last but not least, the study is done with two separate groups of external reviewers. The first group consists of fellow researchers, while the second is made up of practitioners, some of whom also have a research background. The main benefit of having multiple independent groups of reviewers is that it provides further validity and reliability to the outcomes obtained as a result of the study. Moreover, every single person involved in the procedure has sufficient knowledge and practical experience in the subject area (JavaScript), so as to ensure that the participants can be considered as “experts” and that their feedback is indeed meaningful.

C.3.2 Participant roles

There are several people involved in the evaluation process, who have different responsibilities depending on their role. This section provides an overview of the various participants, along with the specific obligations that they have.
Researcher A
The researcher is supposed to design, administer and analyze the results of the evaluation. His duties include:

- Preparing the evaluation protocol
- Planning the review sessions
- Contacting the external reviewers
- Explaining their tasks to them
- Mediating the review sessions
- Examining the results of the sessions
- Drawing conclusions based on the data

Throughout the whole duration of the study, the researcher should attempt to stay unbiased and not influence the outcome in any way, thus ensuring the validity and reliability of the acquired results.

External researcher E1
The role of external researcher E1 is to review the study protocol and make sure that it is well-prepared and that the procedure is designed in such a way so that it minimizes possible research bias. What is more, he should supervise researcher A throughout the entire process and provide help in the form of advice whenever needed. This would guarantee that the study is conducted in a manner which meets high scientific standards.

External researchers
The main duty of each group of external researchers is to analyze the data they are given and in turn provide feedback based on their examination. Their role is to serve as an unbiased external source of knowledge which can provide an impartial opinion regarding the clustering data and the accuracy of the trained model.

C.3.3 Study phases
The evaluation itself is performed in two separate stages. The first one involves individual analysis of the data, while the second consists of a group review session aimed at facilitating discussion among the experts and determining whether they would be able to find common ground regarding their label assignments. The following sections describe each phase in more detail.

C.3.3.1 Individual cluster labeling
The initial step of the evaluation involves contacting the experts (via email) and explaining the goal of both the whole research study and the analysis session in particular, including the
exact procedure and the task they are expected to complete, along with specific instructions on how to do it. During this first phase, the task itself is to analyze the clustering data and assign any label that they deem to be most descriptive of the content within each cluster. The experts are allowed to assign multiple labels to the same cluster as well as assign the same label to more than one of them. Of course, if they are really uncertain regarding a particular group, they can also leave it unlabeled. The data used for the analysis should be formatted in much the same way as it was during the initial pilot testing, with the only exception being the fact that the column headers should be edited so the terms and titles which were extracted at random or using a different technique are not clearly labeled as such. This is done in order to avoid influencing the decisions of the experts, thus further minimizing potential bias.

Once each expert agrees to participate in the evaluation, they are presented with the data and asked to fill it in, following the aforementioned instructions and return it when they are ready. It should be noted that the participants are not given a strict deadline for completing this task which ensures that each individual would have enough time to examine the data and assign an appropriate label, without any predetermined time constraints which could pressure him/her into making a decision more quickly, thus affecting the accuracy of the results.

Finally, after all of the experts are done with their label assignments, a further contact is made in order to settle upon a time and day for carrying out the group follow-up session, at a date which would be mutually agreeable for everyone involved.

C.3.3.2 Group review session

As mentioned, the purpose of the group session is to allow the participants to discuss their choices among each other and see if they can reach a consensus regarding the specific label assignments. Furthermore, this additional seminar will also give them a chance to explain the reasoning behind their decisions, provide clarifications if needed and possibly elaborate further. The session itself consists of two separate parts - a review of the cluster labels given by each expert and a follow-up semi-structured discussion regarding a number of topics including the clustering results, the subject of the study, possible applications of such a solution and so on.

The seminar is started by shortly briefing the participants about the purpose of the session as well as the procedure that is going to be followed. For this reason, a small script is written prior to the meeting so as to ensure that the information is communicated effectively to everyone involved. After that, the experts are presented with an aggregated view of the analysis data, which contains all of their label assignments, along with the labels given during the initial pilot testing. Then, a few minutes are spent on discussing each one of the clusters, by allowing every expert to explain their thought process and rationale for making a particular decision. This is followed by an open dialog among the participants,
which aims to establish a certain level of agreement between them and, as a result, facilitate the selection of a final descriptive label for each cluster.

Once the review of the cluster assignments is done, the next stage of the session is ready to begin. As mentioned, it consists of a semi-structured discussion which will prompt the experts to share their opinions about a wide range of topics. The discussion itself has a very open nature, with only a handful of questions being prepared beforehand which can serve to guide the conversation. However, due to the fact that there is no rigid structure in place, there is a lot more flexibility. For example, if any interesting subject arises during the conversation, additional follow-up questions can be asked, which allows the discussion to delve deeper into the matter.

C.4 Limitations

This section outlines some of the inherent shortcomings resulting from the chosen design of the expert analysis and discusses their effects on the expected outcomes of this evaluation.

C.4.1 Data

As mentioned in Section C.2, the data shown to the participants represents only a portion of the data generated during the feature clustering phase. Therefore, it would not be possible to make any definitive conclusions regarding the accuracy of the clustering model based solely on the expert analysis, because the information that is being examined does not constitute a comprehensive depiction of the actual clustering results. In other words, even after the analysis, it would still be uncertain whether the 30 clusters made by the trained model represent a valid and meaningful set of categories, however, the expert feedback can still help validate the clustering algorithm and the potential usefulness of the solution. Furthermore, since the purpose of the evaluation is to serve as an initial indication of whether the model can accomplish its primary goal (even if it happens to be at a very low level), absolute precision is not required. Another limitation comes from the fact that, as mentioned in Section 6.1.2, the terms and the titles extracted from each cluster cannot be entirely representative of its contents and as a result, it is possible for an incorrect label to be generated. However, this problem can be addressed by repeating the labeling procedure with a different set of data gathered from these very same clusters.

C.4.2 Study design

Another factor which can have an adverse effect on the evaluation is its chosen procedure. Despite the fact that the study is designed in such a way so as to minimize potential biases, they are not eliminated altogether. For example, the number of external reviewers which are supposed to take part in the study may not be enough to ensure that the eventual results are entirely objective, because when using a relatively small sample size, a certain
level of subjectivity would still be present. On the other hand however, having too many participants in the study, may limit their involvement in the discussions. Therefore, it is important to identify the right number of people that should take part in the procedure. Furthermore, since the evaluation is of a qualitative nature, the outcome is highly dependent upon the chosen participants.

C.4.3 Findings

Due to the effects of the aforementioned limitations, the findings of the study will have a reduced generalizability. Moreover, in order to make sure that the obtained results are reliable, the evaluation would have to be repeated using a different dataset as well as additional groups of external reviewers. Nevertheless, given the primary purpose of the analysis, these concerns, while still very pertinent, are not of utmost importance.

C.5 Discussion questions

This section presents the questions which are used to guide the discussion during the second portion of the group review session. These questions aim to provide a general outline that the conversation could follow and serve as a reminder of the specific topics which could be touched upon.

C.5.1 Analysis

Q1: Did you have any troubles with the label assignment?
Q2: Did you find any part of the data to be particularly helpful?

The goal with these questions is to discover if the reviewers had difficulties completing the task and determine what might have caused them. At the same time, the questions can also help establish if the experts found some part of the data to be especially useful for coming up with a suitable label.

C.5.2 Research

Q3: What do you think about the topic of the research?
Q4: What do you think about this approach of finding issue categories?
Q5: What are the benefits and drawbacks of such a method? What are some of the challenges?

The questions shown above are meant to get the experts’ feedback regarding the subject of the thesis and the specific approach that was chosen in order to address the problem at hand. They are encouraged to share their thoughts about the advantages and possible disadvantages that such a technique might entail.
C.5.3 Solution

Q6: Do you think there is potential in this solution or in such a solution in general?
Q7: Do you think such a technology has other possible uses, related to programming issues or otherwise? Can you give an example?
Q8: Would you be interested in trying out a web prototype, based on the same principle, which tries to provide solution suggestions to different issues?

The purpose of these inquiries is to determine whether the participants view the solution as a viable prospect for effectively solving a particular real world problem. What is more, the questions also attempt to find out if they see other possible applications for such an artifact, be it in a similar or a wholly different context.

C.5.4 Evaluation

Q9: As fellow researchers, what are your thoughts regarding the procedure and execution of this evaluation?
Q10: Would you have done something differently?

Finally, the last set of questions allow the participants to share their thoughts regarding the evaluation procedure itself. Since all of them are more experienced researchers, they would be able to provide constructive critique that could be used for identifying possible mistakes, which would in turn help improve future evaluation studies.
Appendix D Expert analysis results

This section presents the results obtained from the expert analysis sessions. The insights acquired from each stage of the procedure, namely the initial label assignment, group review and the subsequent semi-structured discussion are divided accordingly in the following passages. It should be noted that sections D.1-D.3 present the results attained from the first group of experts, while Section D.4 outlines the results gathered from the second one.

D.1 Individual labeling

As explained in Section C.3.3.1, the participants were initially asked to label the clustering data individually. Table D.1 presents an aggregated view of the labels provided by each expert, along with the labeling done during the pilot testing phase. The results illustrate an apparent agreement among the experts that also closely corresponds to the assignments produced via the initial cluster analysis.

Table D.1: The individual label assignments of the analyzed clusters provided by the first group of expert reviewers, juxtaposed with the labels given during the pilot testing of the clustering data

<table>
<thead>
<tr>
<th>Cluster index</th>
<th>Pilot testing</th>
<th>Expert 1</th>
<th>Expert 2</th>
<th>Expert 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>URL Routing</td>
<td>routes, routing, navigation</td>
<td>Routing/application end points URI’s</td>
<td>routing, node.js, angular</td>
</tr>
<tr>
<td>2</td>
<td>DOM</td>
<td>dom, dom manipulation, dom elements, GUI</td>
<td>html markup / html tags</td>
<td>DOM manipulation, UI, GUI</td>
</tr>
<tr>
<td>3</td>
<td>Events</td>
<td>events, callbacks</td>
<td>Event handling errors/troubles</td>
<td>event handling</td>
</tr>
<tr>
<td>4</td>
<td>Return values</td>
<td>return value, functions</td>
<td>Returned unexpected value/return statement</td>
<td>return value</td>
</tr>
<tr>
<td>5</td>
<td>npm, Packages</td>
<td>npm, package management, package manager, setup</td>
<td>Npm configuration/installation of npm packages problems</td>
<td>package manager, dependencies, version</td>
</tr>
<tr>
<td>8</td>
<td>Files</td>
<td>files, file management, resources</td>
<td>File not found exception/file does not exist exception</td>
<td>file, filesystem</td>
</tr>
</tbody>
</table>

The assignments of all the experts can be found here - [https://drive.google.com/open?id=0BwdNIKiwHxCtYm1PR1BGc1d1k](https://drive.google.com/open?id=0BwdNIKiwHxCtYm1PR1BGc1d1k)
<table>
<thead>
<tr>
<th>Cluster index</th>
<th>Pilot testing</th>
<th>Expert 1</th>
<th>Expert 2</th>
<th>Expert 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Testing</td>
<td>testing, software tests</td>
<td>Testing issues/problems/errors</td>
<td>testing</td>
</tr>
<tr>
<td>21</td>
<td>Compatibility</td>
<td>compatibility, resources</td>
<td>Supporting features data type, expressions, library, technology, system, etc.</td>
<td>support</td>
</tr>
<tr>
<td>22</td>
<td>Documentation</td>
<td>documentation</td>
<td>documentation</td>
<td>documentation</td>
</tr>
<tr>
<td>23</td>
<td>Modules</td>
<td>module, package management, package manager</td>
<td>Javascript modules issues/problems</td>
<td>module</td>
</tr>
<tr>
<td>25</td>
<td>Visualization, charts</td>
<td>visualisation, data visualisation, diagrams, graphs, charts, GUI</td>
<td>Charts visualisation issues</td>
<td>alignment</td>
</tr>
<tr>
<td>26</td>
<td>Object properties</td>
<td>properties, variables, assignment</td>
<td>Object properties errors/undefined</td>
<td>(variable) type</td>
</tr>
<tr>
<td>27</td>
<td>npm, Packages, Package installation</td>
<td>npm, package management, package manager, setup</td>
<td>Installation of npm modules issues/errors</td>
<td>npm, package manager, dependencies</td>
</tr>
<tr>
<td>28</td>
<td>JS Functions</td>
<td>functions</td>
<td>Function invocation problems/issues/errors</td>
<td>function</td>
</tr>
<tr>
<td>29</td>
<td>Components</td>
<td>components</td>
<td>Component usage problems/issues/errors</td>
<td>component</td>
</tr>
</tbody>
</table>

Even though, sometimes the participants seemed to use different terms for the labeling, the underlying concept is nearly identical for the majority of the clusters. Moreover, for many of the categories, the reviewers provided several labels, which gives a better idea of their thought process and the way they perceived each cluster. Furthermore, only on a single occasion one of the experts left a cluster unlabeled, presumably because the person could not think of a suitable title. Overall, at this stage, the results appear to be very promising, something which the group review session will give further insights into.
D.2 Group review

As mentioned earlier, the group review\textsuperscript{37} of the individual label assignments aimed to facilitate the discussion among the experts and urge them to find a consensus in choosing a common label for each of the categories. What is more, the review also meant to establish their confidence level regarding the suitability of the chosen label using the following questions:

\textbf{Q1:} \textit{How confident are you that the chosen label accurately represents the data within the cluster?}

\textbf{Q2:} \textit{How confident are you that the label can be used to describe each issue in the cluster?}

The experts were asked to address both questions for each of the analyzed clusters and provide an answer using a scale from 1 to 5, with 1 signifying being “not confident at all”, while 5 indicated being “very confident”. As Lazar, Feng and Hochheiser state (2010, p.282), this sort of coding procedure allows for controlling the \textit{“impact of subjective interpretation”}. Table D.2 shows the final label chosen for each cluster, along with the respective average confidence level of the three experts on both of the aforementioned questions.

Table D.2: The final label assignments of the three experts (Group 1) given after the group review session, along with their confidence levels with regards to the suitability of the chosen labels

<table>
<thead>
<tr>
<th>Cluster index</th>
<th>Final label</th>
<th>Q1</th>
<th>Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Routing</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>2</td>
<td>DOM manipulation</td>
<td>5.00</td>
<td>4.83</td>
</tr>
<tr>
<td>3</td>
<td>Event handling</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>4</td>
<td>Return value</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>5</td>
<td>npm</td>
<td>3.67</td>
<td>4.83</td>
</tr>
<tr>
<td>8</td>
<td>File management</td>
<td>3.33</td>
<td>3.67</td>
</tr>
<tr>
<td>9</td>
<td>Automated testing</td>
<td>4.33</td>
<td>4.00</td>
</tr>
<tr>
<td>21</td>
<td>Compatibility</td>
<td>3.33</td>
<td>3.83</td>
</tr>
<tr>
<td>22</td>
<td>Documentation</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>23</td>
<td>Modules</td>
<td>3.67</td>
<td>4.33</td>
</tr>
<tr>
<td>25</td>
<td>GUI</td>
<td>2.67</td>
<td>2.67</td>
</tr>
<tr>
<td>26</td>
<td>Object properties</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>27</td>
<td>Package installation</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>28</td>
<td>Functions</td>
<td>4.67</td>
<td>5.00</td>
</tr>
</tbody>
</table>

\textsuperscript{37} Full audio recording of the session can be found here: \url{https://connect.sunet.se/p5o6dfu9063/}
Appendix D. Expert analysis results

<table>
<thead>
<tr>
<th>Cluster index</th>
<th>Final label</th>
<th>Q1</th>
<th>Q2</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>Components</td>
<td>5.00</td>
<td>5.00</td>
</tr>
</tbody>
</table>

The data shows that most of the labels which were agreed upon by the experts, correspond to their own assignments given during the individual labeling presented in Table D.1. As a result, with the exception of a few of the labels, the reviewers had a high level of confidence regarding the final choice, which illustrates that they felt it represented the contents of its respective cluster quite well. Figure D.1 shows the aggregated expert confidence levels with regards to Q1 and Q2.

Figure D.1: A diagram showing an aggregation of the expert confidence levels regarding the chosen cluster labels (the dark grey lines signify the standard deviation)

As the diagram illustrates, both questions received a high confidence score, with Q1 getting 4.38 (out of 5) and Q2 - 4.54 respectively. Furthermore, the standard deviation, also shown in Figure D.1 through the dark grey lines, was relatively low for both, namely 0.82 for Q1 and 0.71 for Q2. Even though these results cannot be used to claim any statistical significance, they still demonstrate that there was a consistency of opinion among the reviewers and that the final cluster labels were viewed favorably among them.

Nevertheless, deciding on a specific label for some of the categories sparked a lot of debate, which is reflected through the lower confidence figures in Table D.2. This means that the issues within them might still be a little too dissimilar to fall under a single title or potentially that the chosen label was not the most accurate one (or both).

However, since the experts had a high confidence with over 2/3 of the selected labels, this confirms hypothesis EH1.1, thus demonstrating that the clustering did manage to separate the issues into groups with a clearly identifiable common theme, which in turn answers ERQ1. What is more, the high confidence levels, with regards to Q2, also serve as evidence
confirming hypothesis EH2.1 i.e. that the resulting clusters are, for the most part, self-
contained and hold a distinct issue type. This insight also helps address ERQ2, which
together with ERQ1 provide considerable proof in relation to one of the main research
questions of the entire research (RQ1). Nevertheless, due to the relatively small sample size
and the qualitative nature of the evaluation, no absolute conclusions can be made based
on the acquired results, which provide merely indications regarding the accuracy of the
clustering that would have to be further validated through additional research.

D.3 Post-review discussion

As pointed out in Section C.3.3, the expert discussion following the review of the cluster
labeling was intended to encourage the participants to share their opinion about different
aspects of the research, which would serve to provide a critical outside perspective on
the work that is being carried out. Since the conversation was guided by the questions
outlined in Section C.5, the answers that were obtained, are also divided in a similar fashion,
according to the topic which was being discussed.

D.3.1 Analysis

According to their feedback, the experts found the label assignment task to be anything
but trivial. There were several factors which affected their decision-making process and
made it difficult to come up with an appropriate description. The main trouble arose from
the fact that some portions of the data did not seem to help identify the cluster category
and sometimes actually caused further confusion. This contributed to the initial impression
that the groups did not have a common theme, which necessitated repeated examination.
However, the consensus appeared to be that the differential terms i.e. terms which had the
most significant impact on the eventual assignment of a given issue, were the most helpful
for deciding on a specific label, followed by the full issue titles. This lead to a discussion
regarding the use of terms versus titles for representing the content of a given cluster. On
one hand, the terms provide a more encompassing view of the cluster, since they can be
present in a large number of issues, but lack any sort of context. On the other, titles supply
the necessary context but due to the fact that a full title depicts only a single entry, they
essentially represent a very small sample of the data within the cluster. Of course, these
same considerations were taken into account while determining how to present the cluster
data so that it can be as accurate as possible as well as relatively easy to analyze. The
experts agreed that this is indeed a trade-off and that the chosen format addressed the issue
quite well, but might not necessarily be the most optimal one.
D.3.2 Research

Even though, as they themselves stated, the participants were not very knowledgeable in the domain of natural language processing and machine learning, they still found the research topic to be quite interesting. They thought that the chosen approach for identifying issue categories can work on a high level, but is highly dependent on the quality of the original posts. So if the submitted issue is not explained particularly well, that can have a significant impact on the category assignment and thus result in a wrong classification. Furthermore, due to the fact that the people who report the issues are, for the most part, less experienced, they might use the wrong terminology to describe their problems which would make it even more difficult to determine the correct categories that should be assigned. Therefore, it can be said that the expert feedback confirmed some of the author’s prior apprehensions regarding the design of the proposed solution, mentioned in Chapter 7.

Another observation that the experts made, based on the data they had at their disposal, was that some of the labels appeared to describe the issues within a given cluster quite well, while others seemed to be rather vague and as such, would not be particularly useful from a user perspective, especially if said user is a developer working on the project. Therefore, these categories might require to be further divided into one or more sublevels in order to more accurately represent the specific issue types. As described in Section 5.3.1, this sort of hierarchical category structure was implemented to address the problem of finding similar issues, meaning that such sublevels already exist. However, with each sublevel, the number of categories grows exponentially and so manually labeling them becomes virtually impossible. Furthermore, the subclustering is performed on a dataset that is already divided in some way, so if this division does not represent the natural distribution of the data, any further separation would be biased. Nevertheless, one possible way to address the sheer number of clusters which require labeling, is to harness the expertise of the community, through the use of collaborative tagging, as explained in Section 8.3. This process would allow all of these subcategories to receive a suitable label, without the need for any one single person to analyze an enormous amount of data.

The discussion also revealed that one of the experts thought the resulting categories, signified by their respective label, did not reflect actual problems that users face when working with JavaScript, citing his own experience. However, opinions were slightly divided on the matter, with one of the other participants stating that some of the categories do in fact describe problems experienced during her own work, especially if the labels are taken in the more specific context of a given library or framework. Since each project repository on GitHub has its own dedicated section for issues, any entry posted there is automatically assumed to concern the project itself, thus giving it an extra degree of context, which to some extent alleviates the aforementioned problem.
D.3.3 Solution

The experts stated that they liked the proposed solution, at least based on the results with which they were presented and that they certainly see potential in such a technology. However, one of the concerns they had, derived from the fact that the categories had a varying level of granularity. Some of them seemed to represent a very generic and abstract issue type, while others were rather specific and, in some instances, even obscure, concerning a very particular kind of issue. Therefore, the participants believed that the applicability of these categories in a real world context is unclear and is something that would require further investigation. Nevertheless, they all agreed that, if perfected, such a solution would have a multitude of possible applications, especially when it comes to automatic tagging. For example, the categorization can be extremely helpful on many different forums and Q&A sites like StackOverflow. Since the latter already allows users to manually add tags to their questions when they are posting them, the proposed solution can either serve to automate this process or possibly correct the user-submitted ones, if they happen to be inaccurate.

D.3.4 Evaluation

Finally, the last portion of the discussion was intended to gather the thoughts of the participants about the way the evaluation itself was executed. According to their feedback, none of them had any major issues with the procedure, apart from the fact that the initial description of the task was not clear enough, which caused some confusion regarding what they were actually supposed to do. The experts had differing opinions about the amount of information that was provided - one of them felt that a more detailed explanation about the data itself and what was expected of them, would have helped him understand the task better, while another thought that the “wall of text” they were presented with, was already too long and intimidating. This means that when performing evaluations involving users, it is important to find a balance between having sufficient amount of information regarding a given task and providing a clear and concise explanation of their duties, which does not overwhelm and/or confuse the participants.

D.4 Follow-up analysis session

As mentioned earlier, the second expert analysis session was conducted with another independent group of external reviewers, who expressed interest in the proposed solution. However, since this researcher was not in direct contact with them, the session was largely self-administered, while still following the study protocol outlined in Appendix C. Again, there were three reviewers, but this time instead of researchers, they were practitioners in several different fields, with one of the participants at the same time also being the supervisor of the author. Nevertheless, since he had not seen the results of the original
Appendix D. Expert analysis results

Expert analysis session, prior to making his own label assignments, he was not biased by it. Moreover, since he was intimately familiar with the study procedure, he could mediate the session to a certain extent and ensure that the protocol was indeed followed. The collected data was later submitted to the author of this thesis in order to consolidate the results with the rest of the study.

D.4.1 Participant motivation

The main reason why these practitioners were interested in the system was that both of them utilize a similar technology in their respective area of work, but are not satisfied with the results they are getting. For example, Participant 1 who is a head of quality assurance in a firm which develops software for pharmaceutical companies, is looking for a system which can be used for effective issue triaging i.e. automatically assigning a given issue to the developers who could and should fix it. In the past, this was done manually, by having personnel go through the issues, read them and direct them to the appropriate development team. However, since this procedure is both time-consuming and expensive, they have been looking for a solution which can help them automate this task. Unfortunately, due to the fact that the product they are currently using (provided by HP) is not categorizing the issues as expected, they still need to repeat that manual process. Furthermore, since as mentioned, manually assigning the issue to the correct developers is very time-consuming, it can take up to several days until it is correctly assigned, let alone until it is finally resolved. From a user perspective, this means that it can take quite a long time for a given problem to be solved. Last but not least, often times the “issue” itself is not caused by some incorrect or unexpected behavior on the part of the software, but is instead caused by the fact that the user did not understand how to correctly use the product. Since such cases are easily resolved by providing users with the needed information and due to the fact that they occur on a regular basis, Participant 1 and his company are looking for a system which could provide automatic feedback so that users learn how to fix such trivial problems by themselves. This would serve to speed up the process for both the end users and the developers themselves, who can instead spend their time and efforts focusing on the actual issues that need to be addressed.

On the other hand, Participant 2, who is a senior manager in a large IT consulting company, is working on a system which could identify issues “on the fly”, before users have even reported them. For instance, if the system analyzes the information found in log files and finds a stack trace which specifies that an error occurred, it should be able to recognize that. For this purpose, they need a solution which knows how an issue looks like and can identify such problems as they are happening. Currently, they are using the natural language processing and semantic analytics modules provided by IBM’s Watson, but are also not completely satisfied by the end results, which is why both Participant 1 and Participant 2 were eager to learn more about the proposed solution and see if it has
the potential to help them with their own problems.

D.4.2 Results

Table D.3 shows the results obtained from the individual labeling task done by the second group of reviewers, along with the labels given during the initial pilot testing of the clustering data. As the assignments illustrate, with the exception of a handful of categories, there is an apparent agreement among the experts, as was the case in the prior analysis session. Most importantly though, the vocabulary that they used to describe the various clusters also appears to overlap with the one used by the previous group of experts.

Table D.3: The individual label assignments of the analyzed clusters provided by the second group of expert reviewers, juxtaposed with the labels given during the pilot testing of the clustering data

<table>
<thead>
<tr>
<th>Cluster index</th>
<th>Pilot testing</th>
<th>Expert 1</th>
<th>Expert 2</th>
<th>Expert 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>URL Routing</td>
<td>route, path</td>
<td>route, path</td>
<td>route</td>
</tr>
<tr>
<td>2</td>
<td>DOM</td>
<td>element</td>
<td>wrap, dom</td>
<td>dom, elements</td>
</tr>
<tr>
<td>3</td>
<td>Events</td>
<td>call, fire</td>
<td>event, call</td>
<td>event handler</td>
</tr>
<tr>
<td>4</td>
<td>Return values</td>
<td>return</td>
<td>expected, return</td>
<td>return null or empty</td>
</tr>
<tr>
<td>5</td>
<td>npm, Packages</td>
<td>npm, version</td>
<td>npm, install, package</td>
<td>npm packaging</td>
</tr>
<tr>
<td>8</td>
<td>Files</td>
<td>file</td>
<td>file, json</td>
<td>file missing/incorrect path</td>
</tr>
<tr>
<td>9</td>
<td>Testing</td>
<td>test</td>
<td>test, test case, assert</td>
<td>software test</td>
</tr>
<tr>
<td>21</td>
<td>Compatibility</td>
<td>support</td>
<td>support, help</td>
<td>support of external libraries</td>
</tr>
<tr>
<td>22</td>
<td>Documentation</td>
<td>docs</td>
<td>doc, documentation</td>
<td>documentation incorrect</td>
</tr>
<tr>
<td>23</td>
<td>Modules</td>
<td>module</td>
<td>module, compatibility</td>
<td>module</td>
</tr>
<tr>
<td>25</td>
<td>Visualization, charts</td>
<td>chart</td>
<td>chart, bar</td>
<td>charts</td>
</tr>
<tr>
<td>26</td>
<td>Object properties</td>
<td>undefined</td>
<td>read, read property</td>
<td>property of undefined/null</td>
</tr>
<tr>
<td>27</td>
<td>npm, Packages, Package installation</td>
<td>npm, install</td>
<td>npm, install, package</td>
<td>npm install</td>
</tr>
<tr>
<td>28</td>
<td>JS Functions</td>
<td>function, work</td>
<td>function, expected, return</td>
<td>function errors</td>
</tr>
<tr>
<td>29</td>
<td>Components</td>
<td>component</td>
<td>component, reload</td>
<td>component</td>
</tr>
</tbody>
</table>
Appendix D. Expert analysis results

Again, the experts have taken advantage of the fact that they are allowed to assign multiple labels to each category, which enables them to better communicate their ideas. Furthermore, on this occasion, none of the participants have left a category without a label, which most likely signifies the fact that they were more inclined to take a chance with their assignment than leave an empty space. Besides, in rare instances, they have decided to use full expressions to represent the cluster label, which is a strategy that was not adopted by the previous group of reviewers.

Naturally, the individual assignment was followed by a group review session, where the experts could discuss their choices, settle upon a final label and declare how confident they feel with their selection. Table D.4 presents the results of this session, along with the final label chosen during the prior expert analysis study, for comparison purposes.

Table D.4: The final label assignments of the three experts (Group 2) given after the group review session, along with their confidence levels with regards to the suitability of the chosen labels

<table>
<thead>
<tr>
<th>Cluster index</th>
<th>Previous session</th>
<th>Final labels</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Routing</td>
<td>route, path</td>
<td>4.5</td>
</tr>
<tr>
<td>2</td>
<td>DOM manipulation</td>
<td>dom, element</td>
<td>4.25</td>
</tr>
<tr>
<td>3</td>
<td>Event handling</td>
<td>call, event</td>
<td>4.25</td>
</tr>
<tr>
<td>4</td>
<td>Return value</td>
<td>undesired return</td>
<td>4.25</td>
</tr>
<tr>
<td>5</td>
<td>npm</td>
<td>npm</td>
<td>5.00</td>
</tr>
<tr>
<td>8</td>
<td>File management</td>
<td>file</td>
<td>5.00</td>
</tr>
<tr>
<td>9</td>
<td>Automated testing</td>
<td>test, test case</td>
<td>5.00</td>
</tr>
<tr>
<td>21</td>
<td>Compatibility</td>
<td>support</td>
<td>4.25</td>
</tr>
<tr>
<td>22</td>
<td>Documentation</td>
<td>documentation</td>
<td>4.20</td>
</tr>
<tr>
<td>23</td>
<td>Modules</td>
<td>module</td>
<td>4.75</td>
</tr>
<tr>
<td>25</td>
<td>GUI</td>
<td>chart</td>
<td>4.5</td>
</tr>
<tr>
<td>26</td>
<td>Object properties</td>
<td>read property</td>
<td>4.25</td>
</tr>
<tr>
<td>27</td>
<td>Package installation</td>
<td>npm install</td>
<td>4.5</td>
</tr>
<tr>
<td>28</td>
<td>Functions</td>
<td>function</td>
<td>4.5</td>
</tr>
<tr>
<td>29</td>
<td>Components</td>
<td>component</td>
<td>4.5</td>
</tr>
</tbody>
</table>

As the data shows, there are slight dissimilarities in the results that were obtained compared to the previous analysis session, considering both the chosen labels and the confidence level of the reviewers. Nevertheless, the study results clearly demonstrate that the experts had a very high confidence regarding their choices, which for the most part, overlapped with the selection made by the initial group of reviewers.

Finally, during the post-review discussion, both of the practitioners involved in the study, expressed their belief that the proposed solution would be useful for their respective field of work and that, even though it is still in a very early phase of its development,
it is going in the right direction. What is more, in their opinion, some of the potential benefits that this kind of solution can provide to them, include having a better overview of the status of the projects they are currently involved in, getting to know which are the most common types of issues that they are facing at the moment and highlighting where they should focus more training in their respective companies. Besides, they have declared that they would like to know more about the project so they can see how they could adopt something similar in their own work and get guidance for a solution selection and/or an internal implementation.

These confessions serve to largely dispel the reservations that the previous group of experts had about the real world applicability of the categories produced by the proposed solution. Furthermore, the practitioners have suggested a number of other possible applications, for instance in the areas of issue triaging and automatic issue identification. However, despite this particular disagreement between the two groups, a lot of the post-review feedback acquired during both studies seems to overlap. For example, both groups agree about which part of the data was most useful (Section D.3.1), about the fact that the chosen method for identifying categories depends a lot on the quality of the original input (Section D.3.2), about the slightly confusing task description (Section D.3.4) and so on. Nevertheless, the practitioners believe that the system could evolve and learn in the future, so as to be able to overcome most of its current shortcomings.

D.4.3 Analysis

Since two independent groups of reviewers produced largely similar labels for the clusters they were presented with, it can be concluded that the clustering model was able to divide distinct issue types into separate categories, thus fulfilling its main purpose and demonstrating the feasibility of the chosen approach. Moreover, the fact that the system was positively received by people with both a practically-oriented as well as a theoretically-oriented background speaks volumes of its potential. What is more, the difficulties that the practitioners involved in the expert analysis have had to deal with in their line of work, mentioned in Section D.4.1, illustrate that the proposed solution can have a wide range of real life applications and can help address a number of different problems. Besides, such a system would be especially helpful for proprietary software for which, unlike open source technologies, there might be limited information available on the web. Furthermore, since the code base is not freely distributed and usually there is only a limited community around the technology, the burden of providing support and helping solve various issues that users encounter, falls primarily on the shoulders of the developers. Therefore, having a system which can analyze issues and use the acquired information to either categorize them, assign them to a particular group of technicians, automatically suggest a solution to the end user or all of the above, can be immensely beneficial.
Appendix E User evaluation protocol

This section outlines the study protocol prepared for the user evaluation stage of the research. The aim of the evaluation is to determine if the developed web prototype can perform its function and support people in their search for solutions to the various programming issues they encounter. The section introduces the research questions which the study aims to answer and describes the specific procedure that is to be followed throughout the duration of the testing. Lastly, it presents some of the possible limitations of the evaluation and discusses how they affect the outcome of the user sessions.

E.1 Research questions

As mentioned, the goal of the user evaluation is to assess whether the developed web prototype can help users find relevant solutions to their programming issues. Since the prototype primarily relies on the issue analysis performed through the use of various natural language processing and machine learning techniques, the results of the evaluation can serve as means for addressing RQ2 and RQ3, which were formulated as such:

**RQ2:** Which natural language processing and machine learning techniques can be applied to automatically assign an issue to a particular category?

**RQ3:** Which computable approaches can be utilized to identify possible solutions that are relevant for the identified issue categories?

Due to the fact that the workflow for discovering possible solutions is built upon the clustering model trained as part of the issue categorization efforts, the developed web application makes use of the model’s issue assignment capabilities in order to offer relevant suggestions. Therefore, if the user testing demonstrates that the prototype can indeed aid users in their search, this would provide strong evidence towards both of the aforementioned research questions. Even though a preliminary analysis was already performed, which helped uncover some of the shortcomings of the implementation, getting feedback from actual users will provide a much more detailed understanding of how well the prototype can perform its functions. That being said, here are the main research questions (URQx) that the user evaluation will attempt to answer:

**URQ1:** Does the platform provide value for the end users?

**URQ2:** How does the prototype compare to traditional ways of searching for solutions?

The purpose of these questions is to assess the usefulness of the implementation and thus
determine if the underlying model can be utilized in order to accomplish tasks other than issue categorization. If that is to be the case, this would support the notion that the same natural language processing and machine learning techniques which were used for separating issues into categories, can also help solve a different type of problem. What is more, the study will also try to establish how well can the develop prototype compare with the search methods and strategies usually employed by users. This would help examine the matter from a more practical perspective, namely if the implementation can compare favorably with traditional ways of finding solutions. The rationale here is that the prototype should provide some concrete advantage, if people are to use it instead of relying on other tools and techniques. However, since the term “value” mentioned in URQ1, is a bit vague and can encompass a wide range of possible benefits, the question can be broken down into a few more specific questions, such as:

**URQ1.1:** *Can the prototype aid users in their search for solutions to programming issues?*

**URQ1.2:** *Can users find relevant information using the application?*

**URQ1.3:** *Are they able to discover something interesting or learn something new?*

Due to their more narrow focus, these questions are better suited for evaluating specific aspects of the implementation, which in turn helps address URQ1 and thus also RQ2 and RQ3. Naturally, the information needed to answer them will be acquired through the feedback of the users involved in the testing sessions. Therefore, the sessions should be designed in such a way so as to facilitate this process. Based on URQ1 and URQ2, here are the different hypotheses which can be formed:

**UH1:** *The prototype provides some sort of value to the users*

If the participants find that using the implementation benefits them in some way, this would serve to demonstrate the feasibility of the chosen approach and would emphasize its relevance not only from a research perspective, but from a practical one as well.

**UH2.1:** *The prototype is more useful than traditional search methods*

In case the prototype is perceived to be more useful than the usual search methods employed by users, this would provide further support towards its real world applicability, because it would indicate that people might have a clear incentive to use it.
Appendix E. User evaluation protocol

**UH2.2: The prototype is as useful as traditional search methods**

If the application is deemed to be on the same level as traditional search techniques with regards to usefulness, this would show that, even in its current form, it can be a suitable alternative to the usual means of searching for software related solutions.

**UH2.3: The prototype is not as useful as traditional search methods**

Finally, if the evaluation results indicate that users do not believe the prototype to be as useful as regular methods of searching, this would show either that the implementation is not yet ready for exploitation or that the developed workflow itself might not be well-suited for addressing this sort of task. Moreover, this could also mean that the implementation might have a limited practical application, due to the fact that, it would be difficult to convince people to adopt it if it does not offer a tangible advantage over traditional alternatives.

**E.2 Procedure**

This section presents the specific procedure of the user evaluation sessions. It describes the overall study design as well as the different types of roles adopted by the people involved in the process, and outlines the various stages of the assessment itself.

**E.2.1 Study design**

In order to reduce potential research bias, a number of precautions have been taken in designing the study procedure. First and foremost, the study protocol has been thoroughly reviewed by an external researcher so as to ensure that it has been properly thought-out and that it is up to the required scientific standard. Furthermore, the participants chosen to take part in the evaluation have not been previously involved in any capacity during the development of the project, meaning that they come into the testing sessions with no prior knowledge of the system, which could have influenced their perception of it. What is more, all of the participants have been selected on the basis that they have some domain knowledge and experience in the area of JavaScript, so as to ensure that their feedback is relevant to the study. However, the exact knowledge level varies quite a bit depending on the person, which makes the selected sample of users more diverse and in some respects more helpful given their distinct points of view, despite the relatively small number of people involved. Moreover, the participants represent a wide range of users with both practical and theoretical backgrounds, including industry professionals and university students. Lastly, the issues which are being used for testing the capabilities of the prototype are chosen by the study subjects themselves and they are also the ones who determine whether the presented
results are relevant to their search, thus eliminating any bias that could come from the researcher administering the session.

### E.2.2 Participant roles

There are a number of different people taking part in the overall evaluation process, each with their own set of unique responsibilities. This section presents an overview of the various participants involved in the user testing, along with the specific duties they have according to their assigned role.

#### Researcher A

The main responsibilities of the researcher are to plan and conduct the evaluation as well as reflect upon the results obtained through it. Some of his other duties include recruiting the study participants, familiarizing them with the procedure that is to be followed and mediating the evaluation sessions themselves. Throughout the entire process, the researcher needs to stay impartial and not influence any of the participants, and in general not compromise the evaluation results in any way.

#### External researcher E1

External research E1 is responsible for reviewing the evaluation protocol and identifying possible weak points in the design, which should be addressed in order to improve the study procedure and thus lessen the adverse effects of research bias. Furthermore, he should also advise researcher A and provide constant feedback regarding the various design decisions that researcher A makes so as to ensure that the evaluation study is conducted in accordance to established scientific standards.

#### User evaluation participants

The participants of the user evaluation sessions are the ones who will be testing the web prototype and giving their feedback based on their experience. Their role is to perform several tasks which are designed to assess different aspects of the implementation and then share their comments and remarks, which will help address the research questions outlined in [Section E.1](#). The participants themselves should be representative of the potential users of the application, which encompasses a rather large group, including developers working in the industry, researchers, students and even programming enthusiasts.

### E.2.3 Study phases

The user evaluation consists of two main stages - an initial pre-test activity where participants are asked to complete a given task, and the testing session itself, where they use the web application and provide feedback regarding their experience. Both of these phases are
Appendix E. User evaluation protocol

presented more thoroughly in the following paragraphs. It should be pointed out that the entire evaluation is conducted on an individual basis with each user involved in the study.

E.2.3.1 Pre-evaluation activity

The first phase of the study is performed prior to the actual evaluation session and it involves each participant (individually) describing in a sentence, each of the last 5 JavaScript related issues that he/she have had to overcome. The issues themselves can be of any kind, be it very specific or very general, and concern any library or framework, as long as the programming language in question is JavaScript. Since it might be difficult for people to remember particular problems that they had to deal with, they are allowed to go through their browsing history and examine their previous search queries in order to refresh their memory. The reason for gathering this information is so that these same issues could be used for the subsequent prototype evaluation. As mentioned, the benefit of this approach is not only the fact that it eliminates any bias on the part of the researcher, but also ensures that the participants are able to more objectively assess the results that are being presented to them, because presumably they have already found a solution to each of these issues and are therefore better equipped to make an informed judgement. So, once a participant has prepared a written description of these 5 issues and has submitted it to the researcher, the next stage of the evaluation can commence.

E.2.3.2 Evaluation session

The second and final phase of the user evaluation study is when the users are able to interact with the prototype and share their thoughts regarding their experience. At the start of the session, the participants are first familiarized with the procedure that is to be followed using a previously prepared script which ensures that the briefing they receive is consistent. Then, they are presented with a small questionnaire which aims to gather some information about their background, experience with JavaScript and general search tendencies. This data will serve to provide additional context to the post-test feedback, which will help to better understand their answers during the subsequent analysis.

After users complete the initial questionnaire, they are finally introduced to the actual web application and its functionality as well as the output that it produces. Then, they are given the 5 JavaScript issues which they themselves identified during the pre-evaluation activity and asked to use the prototype and its various capabilities to try and find a solution to each issue. The participants are encouraged to experiment with the application and dig into the results as much as they would like, until they feel they have found a satisfactory answer or have given up on their efforts to do so. Furthermore, users are also requested to give an estimation of how relevant the discovered solution is, using a scale from 1 to 10. Also, if they feel like they might be able to find a better alternative, they could continue
on with their search. However, in order to regulate the duration of the session and keep it from becoming overly long, they are given a maximum of 5 minutes of exploration per issue. They are also allowed to follow external links and examine the content that is being presented there, but are restricted from using traditional search engines. Furthermore, users can also completely change or slightly adjust their search queries if they would like, in case they feel that it would affect the output they are getting. Whenever the participants feel that they have managed to find an answer which they consider to be a relevant solution to their problem, they can communicate that to the researcher and move on to the next issue. If they are unable to do so or if the allocated time has run out, they would also move on and repeat the same process with the next issue. It is important to point out that since the entire session takes place online, the participants are also asked to share their screens, so that the researcher can observe how they interact with the prototype and take notes throughout the duration of the testing session. This not only provides additional context to their questionnaire answers, but also allows for comparing what users say to what they actually do (Gray, 2004 p.238). Once the participants have gone through the 5 issues, the prototype testing is complete.

For the last part of the evaluation, users are again presented with a short questionnaire, consisting of several statements regarding the prototype. Depending on the level to which they agree or disagree with the statement, they are asked to select a value ranging from 1 to 5, with 1 signifying that they “strongly disagree” and 5 that they “strongly agree” with the given statement. The participants are also welcome to share any additional feedback that they might have as well as to elaborate on their various choices, as they are making them. After completing the questionnaire, the entire evaluation session is over.

E.3 Limitations

This section discusses some of the possible limitations of the user evaluation stage of the research as well as their effects on the overall validity of the study.

E.3.1 Data

Despite the fact that the approach of transferring the task of selecting the specific issues which are to be used for the evaluation to the participants themselves, eliminates any potential bias on the part of the researcher, it also has certain limitations. Since there is no oversight of the selection, it is not clear whether the chosen issues represent the kind of entries which predominantly appear in the “Issues” section of a GitHub repository i.e. genuine programming problems. If that is not the case, it would be difficult for the system to provide a relevant response, due to the fact that the data used for the training of the categorization model would be rather different compared to the submitted search query. However, if it was up to the researcher to assess which entries are acceptable and which are
Appendix E. User evaluation protocol

not, this would again introduce bias. Nevertheless, as Herzig, Just and Zeller (2013) found out bugs are often wrongly classified as such, a problem that based on some initial analysis, also seems to apply to the issues on GitHub. This means that the developed solution might be able to produce relevant output based on a query that may be somewhat unsuitable for the platform, because some of the entries contained in the original dataset, could represent such misclassifications.

### E.3.2 Participants

One of the main shortcomings of the study is the fact that it involves a rather small number of test participants. As a result, the acquired data might not be enough to warrant making certain conclusions regarding the current state of the application. However, due to the fact that the prototype is still at a very early stage in its development and the evaluation itself has a predominantly qualitative character, it is still possible, even when using a small sample, to receive valuable feedback that could help identify potential weaknesses and improve the overall design of the solution.

### E.3.3 Study design

Another limitation of the evaluation procedure that is inherent to the chosen design is the fact that the study relies primarily on qualitative data. As a result, even though the acquired feedback is much more nuanced and detailed, the outcome of the study can vary greatly depending on the specific people involved in the evaluation. Therefore, the testing session would have to be replicated with a different set of users and obtain similar results so as to verify that the data is indeed reliable.

### E.3.4 Findings

As mentioned, due to the qualitative nature of the study and the small sample size that was used, the findings attained through it have a limited generalizability and reliability. As such, they are subject to a certain degree of uncertainty and would have to be supported by further research if they are to be taken at face value. Nevertheless, the evaluation can still provide useful and relevant information that can help assess whether the developed prototype achieves its purpose and determine which aspects of it should be modified in order for it to perform its functions even better.
E.4 Questionnaires

This section presents the questions used for the two questionnaires conducted before\(^{38}\) and after\(^{39}\) the prototype testing session. What is more, it explains the rationale for asking these questions as well as the exact information that is expected to be gained through the eventual answers.

E.4.1 Pre-test questionnaire

Q1: Primary occupation?
Q2: Main areas of expertise?

The purpose of these questions is to gather some information about the professional background of the participants and determine what their main areas of interest and expertise are. This data can help create a more detailed profile of the users and thus add more context to their answers.

Q3: How long have you worked with JavaScript?
Q4: When was the last time you worked with JavaScript?
Q5: How knowledgeable would you say you are in JavaScript?

The aforementioned questions aim to assess the experience and knowledge level of the users, when it comes to JavaScript. This will help establish whether they are novices, intermediates or even experts in working with the technology, which could give more insight into their evaluation of the prototype.

Q6: Which of the following platforms do you use when searching for solutions to programming issues?
Q7: How often do you use these resources for finding solutions?
Q8: How helpful do you find these resources?
Q9: How difficult is it to find the information you are looking for on these platforms?
Q10: How helpful do you find the following types of answers?

The last portion of the pre-test questionnaire intends to help better understand the search habits of the participants as well as find out how favorably they view different types of online resources. On top of that, the last question aims to discover which types of answers are seen as most helpful by the users. This information not only provides a detailed look into the way they search for programming solutions on the web, but can also be utilized for

\(^{38}\) Pre-test questionnaire - [https://goo.gl/forms/lt2z6v5g4nqJrQa32](https://goo.gl/forms/lt2z6v5g4nqJrQa32)

\(^{39}\) Post-test questionnaire - [https://goo.gl/forms/lRzvR0gwd1jG1Lgs1](https://goo.gl/forms/lRzvR0gwd1jG1Lgs1)
determining how to improve the prototype and make the search results more accurate and relevant from a user perspective.

E.4.2 Post-test questionnaire

Q1: The results were relevant to my search
Q2: I was able to find answers to my questions
Q3: I could easily find what I was looking for
Q4: I was able to learn something new
Q5: I discovered something interesting

The statements shown above, attempt to provide an answer to URQ1, by establishing whether the application provides any value to the end users. As mentioned in Section E.1, the term “value” can have a rather broad meaning, and therefore the statements focus on specific benefits that the prototype might bring to the person using it, ranging from presenting relevant answers to facilitating the discovery of new and interesting information.

Q6: The prototype was more helpful than the search methods I normally use
Q7: Such an application would be useful for finding solutions to programming issues

This set of statements focuses on estimating the perceived usefulness of the application. While the first helps address URQ2 and determine if the developed solution is deemed to be more helpful than traditional search methods, the second aims to find out if users have a favorable view of the idea to have such a tool. This would serve as evidence that the prototype could have real world applicability and that people would be willing to adopt it.

Q8: The “precision” setting had an impact on the output
Q9: Which “precision” setting provided the most useful results?
Q10: Which part of the results was most helpful?

Finally, the last portion of the post-test questionnaire is concentrated on the functionality of the prototype itself and how it can potentially be improved. The questions aim to determine three separate things - if the “precision” setting that was implemented has successfully done its job, at which level in the clustering hierarchy did the participants find the most relevant answers and lastly, which part of the output was seen as most helpful. The received feedback can help identify both weaknesses and strengths in the design and thus contribute to building a more complete solution.
Appendix F User evaluation results

This part of the Appendix presents the results acquired during the user evaluation phase of the research. The study itself was conducted with 6 participants, each with a unique background and skillset. The evaluation process helped gather a variety of useful data about the participants and their opinion of the developed prototype, including answers from the questionnaires, observations made during the testing sessions and feedback obtained directly from the users themselves. Therefore, this section also provides an analysis of the study results and their overall contribution toward the research topic.

F.1 Pre-test questionnaire

As mentioned in Section E.2.3.2, the pre-test questionnaire aimed to gather some additional information about the background, general knowledge and search habits of the participants, which could be used to provide more context to their post-test feedback. The first set of questions intended to gather some information regarding the professional background of the participant and experience with JavaScript.

Figure F.1 shows that the majority of the people involved in the study identified themselves primarily as developers. However, many of them hold more than one occupation at the same time, for example, working part-time while studying or vice versa. Unfortunately, this is not reflected in the results because the question demanded a single answer, something that may be seen as a shortcoming of the questionnaire design.

**Primary occupation?** (6 responses)

![Pie chart showing the primary occupations of the participants](image.png)

**Figure F.1:** A diagram illustrating the self-proclaimed primary occupation of the chosen participants
Furthermore, the majority of the participants consider their primary areas of expertise to be related to web development - be it front-end or back-end, as shown in Figure F.2. Therefore, it is virtually impossible for them not to be familiar with JavaScript as a technology, since it is an essential component to any sort of development for the web, especially in recent years.

**Main areas of expertise? (6 responses)**

![Bar chart showing the main areas of expertise (in relation to software development) of the study subjects](image)

As the answers from the next two questions shown in Figure F.3 illustrate, this assumption holds true - all of them have been working with JavaScript for a while and some of them even do so currently or as recently as a week ago (from the time of the study). However, nearly all of the participants are relatively new to the technology, having started working with it less than two years ago.
Appendix F. User evaluation results

How long have you worked with JavaScript? (6 responses)

- Under 1 year: 50%
- 1-2 years: 16.7%
- 3-5 years: 33.3%
- Over 5 years: 16.7%

When was the last time you worked with JavaScript? (6 responses)

- This week: 33.3%
- More than a week ago: 33.3%
- Between 1-3 months ago: 33.3%
- Between 4-6 months ago: 33.3%
- More than 6 months ago: 33.3%

Figure F.3: Charts demonstrating the relative JavaScript experience of the participants as well as their most recent use of the technology

This relative inexperience is also evidenced in the answers they gave to the next question, which asked them to assess their knowledge level regarding the language on a scale from 1 to 10. Most of the participants chose a value that fits within the range of 5-7 (Figure F.4), indicating that they feel they are currently at an intermediate level, with the exception of one study subject who did not appear to be as confident and chose a much lower number.
Appendix F. User evaluation results

Figure F.4: A chart showing the users’ self-assessed JavaScript knowledge level

The next portion of the questionnaire aimed to learn more about the search tendencies of the users and get their opinion about a number of online resources that are generally used for finding solutions to programming issues. As Figure F.5 shows, the most commonly used platforms for this purpose are search engines, Q&A websites (e.g. StackOverflow) and code hosting sites like GitHub, closely followed by reference platforms such as MDN or w3schools.

Figure F.5: An overview of the different online platforms and resources that users utilize in order to search for solutions to programming issues

Surprisingly, not all of the participants appear to make use of official documentation,
which exists specifically to provide answers to the most common inquiries that people might have regarding a given technology. Other popular sources for finding solutions include online articles and tutorials, books and video sharing websites like YouTube.

The next batch of questions was intended so as to find out more about the way users interact with the platforms they chose in the previous entry and understand how they perceive them. Each participant was asked to rate only the platforms selected in the prior question. As Figure F.6 shows, the most often used resources seem to be search engines, Q&A sites, reference platforms and documentation. Code hosting platforms and online articles and tutorials appear to be a less popular choice, while the other options are only utilized by a handful of people.

![Figure F.6: A diagram illustrating how often the participants use various online resources for finding solutions to programming-related problems](image)

When asked to assess how helpful these resources are, again search engines and Q&A websites come out on top, once more followed by reference platforms and documentation, as illustrated in Figure F.7. However, the fact that half of the people who use official documentation, believe that it is only helpful some of the time, may provide an answer to the question why it is not universally relied upon. The participants also had a rather mixed opinion about code hosting platforms, which does not bode particularly well for the developed solution.
Finally, the last question in this group attempted to determine how difficult it is for users to find the information they are looking for when browsing through these resources. Once more, search engines and Q&A sites were the most highly rated entries along with reference platforms (Figure F.8). Documentation again got a relatively mixed score, which is a fact that helps understand the somewhat unfavorable reputation that it has among the participants.
However, code hosting platforms received an even worse rating, with most of the participants stating that the difficulty of finding relevant information is largely dependent upon the situation. Of course, as the data shows, the answers to all three questions presented in Figures F.6-F.8 are intrinsically related - if users can easily find the information they are looking for, that in turn makes the platform appear more helpful which in turn encourages them to use it more often. That is why, resources like search engines, Q&A websites and reference platform consistently get positive feedback, while a code hosting site like GitHub is not viewed as favorably. This might be a little problematic for the study, because it could indicate that users have already made up their mind and decided that GitHub is not a particularly useful source for finding solutions to programming issues. Nevertheless, even if that is indeed the case, if the prototype does manage to help them address their software related problems, this could serve to change their viewpoint on the matter.

Figure F.8: A chart showing the perceived ease of finding relevant information on several online platforms, as indicated by the evaluation subjects
Lastly, the final question of the pre-test survey concerned the content of the actual answers that could be found on these platforms as well as the opinion that the participants had about them. As Figure F.9 demonstrates, the most universally liked option was code samples, which was found to be helpful by everyone involved in the study, followed by detailed explanations and links to external sources, for which the results were very ambiguous. Since people seem to find code snippets as the most useful form of solution to their programming problems, most probably because they can easily be copied and pasted directly into their own code, one potential future improvement of the prototype could be to favor answers containing pieces of code and give them a bigger “weight” when deciding how relevant and valuable a given answer is.

### F.2 Prototype testing

Once the study subjects filled-in the initial survey, the most important phase of the evaluation i.e. the prototype testing, was ready to begin. As explained in Section E.2.3.2, participants were asked to use the 5 JavaScript issues they had identified prior to the testing session and try to find a suitable solution to each of them through the application. This section presents some of the observations that this researcher made during the sessions as well as the direct feedback gathered from the users themselves regarding the prototype they were testing.

#### F.2.1 Observations

Based on the behavior that the participants exhibited during the study, it appeared that the links contained within the suggested solutions, which were pointing to the original source,
Appendix F. User evaluation results

helped users browse through the results more quickly. The reason for this was the fact that due to the way they are structured, the URL’s signified which repository the answer was pointing to, and so users could easily determine whether said repo could provide useful information pertaining to their search. For instance, if the participant was searching for an answer regarding AngularJS and several of the links shown in the results pointed to ReactJS, he would immediately know that they are not relevant to his query.

What is more, users were often frustrated and slightly confused about the fact that adding keywords or changing phrases and word combinations did not have the desired effect on the output. In fact, often times it seemed that the platform did not even take into account parts of their input. Naturally, for someone who is used to traditional search engines, this would not be considered as expected behavior. Moreover, if the query itself was not well-formulated, that decreased the chance of getting a useful response even further.

In order to deal with this problem, some participants chose to tweak the search query as much as they could and then examine the outcome of their actions. Overall, it seemed that people had differing searching strategies and styles - while some were very selective in deciding whether to click on a link or not, others were much more willing to follow the external links, even if they were not entirely sure that they would be able to find any relevant information there. Moreover, some of the participants also seemed to be more inclined to experiment with the “precision” setting of the application. One possible explanation of this phenomenon is that some users adopted a more exploratory approach, where their primary objective was to discover and learn, while others had a more goal-oriented strategy and were mainly focused on accomplishing their goal as quickly and efficiently as possible (as they would, if they were using an actual search engine).

F.2.2 User feedback

All the participants agreed that questions regarding “pure” JavaScript or just about a more general concept in the language are highly unlikely to be found on a platform such as GitHub, where the majority of the issues concern a specific project and are very concrete in nature. This is an inherent limitation of the platform itself which means that it is largely unable to provide answers to a wide range of programming issues that users might encounter. Furthermore, often times the “issues” that people face can be defined as such in the broadest sense of the word, referring to “not knowing how to accomplish a specific task in order to attain a certain outcome”. As the participants themselves stated, they have very rarely found the answers to such inquiries, or any sort of inquiries, on GitHub, even though as the results from the pre-test questionnaire show, all of them have used the platform at one time or another to resolve a programming problem. Instead, when they have a question of such nature, they almost always turn to a Q&A website like StackOverflow. However, since the starting point for their investigation in virtually every instance is a search engine, the fact that StackOverflow shows at the top of the results also contributes to this phenomenon.
Appendix F. User evaluation results

(which is not to say that StackOverflow is not an excellent resource). Nevertheless, one of the participants pointed out that this shortcoming of the GitHub platform necessitates that the query is formulated in a very specific manner in order to increase the likelihood that it provides some sort of a useful output. For example, instead of searching for “how to parse JSON”, which constitutes a general question about the JavaScript language, the query should be rephrased so that it essentially becomes an “issue”. Therefore, this same search string would turn into “Cannot parse JSON”, which is much more likely to yield a helpful answer, even if it may be related to a specific library or framework.

The participants also expressed their view that, even when their search for a specific solution brings them to GitHub, the answers they could find there, are for the most part, not too helpful. Furthermore, as the pre-test questionnaire illustrates, sometimes it is difficult for them to find the information they are looking for on the platform, especially compared to Q&A sites. The main reason for this might be the layout of the webpage, which in the case of GitHub looks like a regular forum thread, where the answers are presented in a descending chronological order, with none of them really standing out and so the visitors may need to go through all the comments in order to find the information they are searching for. On the other hand, on a site like StackOverflow, where finding relevant answers is the number one priority, a number of measures have been taken so as to facilitate this process. For instance, best rated answers appear at the top, answers appear more prominently than comments and so on.

Despite all of this, some users did in fact manage to find some semblance of an answer to several of their questions. Out of the 30 queries made by the 6 participants involved in the study, in 9 cases, the users felt that they managed to find an answer related to their query. This means that the prototype was able to deliver a suitable solution about 1/3 of the time. However, when asked to rate the answer with regards to its relevance and completeness, the average score that the study subjects gave was around 5/10, due to the fact that, as they stated, even though the entry was related to the original question, it did not provide a direct answer to it. Nevertheless, they admitted that this suggestion might serve as a good starting point in their search for a definitive solution. However, 6 of those 9 cases came from only two of the participants and thus it might be argued that they could have had a more generous definition of what actually constitutes an “answer” than the others.

F.3 Post-test questionnaire

After users had the chance to test out the web prototype, they were asked to provide some feedback regarding their experience via another questionnaire. As explained in Section E.4.2, the answers to these questions would help address research questions URQ1 and URQ2, outlined in Section E.1, thus determining whether the developed solution provides value to the end users and if it can be compared to the traditional search methods that they normally utilize.
Appendix F. User evaluation results

Taking into account the outcome of the evaluation itself, it would not be unreasonable to conclude that the user feedback will not be particularly positive, especially when it comes to the primary function of the application - providing relevant answers to programming related queries. As Figures F.10-12 show, the participants more or less unanimously agree that the prototype was unable to fulfill its main purpose. Users felt that the results presented to them were not really relevant to their search (Figure F.10) and that the search process itself was not only difficult (Figure F.12) but it also did not ultimately help them find answers to their questions (Figure F.11).

![The results were relevant to my search](image)

**Figure F.10:** A diagram showing the perceived relevance of the search results produced by the web application prototype

![I was able to find answers to my questions](image)

**Figure F.11:** A chart demonstrating whether users believe they were able to find answers to their inquiries
Figure F.12: A diagram illustrating the perceived ease of retrieving relevant information using the web prototype

However, these results are most likely affected by the number of issues that the system has been trained with and uses for finding solutions. In a future release, a larger dataset should be used in order to establish if the accuracy of the output would satisfy users to a larger extent. Nevertheless, the participants still thought that the application had some positive aspects, in that it helped them learn something new (Figure F.13) and/or discover something interesting (Figure F.14). Therefore, it can be concluded that the prototype did manage to provide value to the users, since they were able to acquire some new knowledge through it, even though, as mentioned earlier, it did not help them meet their primary objective. Furthermore, on the limited number of occasions when users were able to find information pertaining to their problem, they stated that these results could serve as a good starting point toward finding a definitive answer, thus further supporting the claim that even in its current form (which is far from perfect), the prototype can still be useful. This insight serves to confirm hypothesis UH1 and in turn provide an answer to URQ1. Moreover, it also helps demonstrate that, despite some of its flaws, the proposed solution is going in the right direction.
The next set of questions aimed to establish how the participants felt about the solution in general and how they would compare it to the search methods and techniques that they usually rely on. As Figure F.15 shows, users unequivocally thought that the prototype was not as useful as regular search options, which confirms hypothesis UH2.3 and thus answers URQ2. This casts some doubt over the real world applicability of the application, unless it is improved greatly, for instance by increasing the dataset and/or reviewing the clustering techniques so as to identify additional parameters, because otherwise people would not have a sufficient incentive to adopt such a product.
Appendix F. User evaluation results

The prototype was more helpful than the search methods I normally use
(5 responses)

Figure F.15: A chart displaying how helpful the web prototype was considered to be, compared to traditional search methods normally used by the participants

However, the answers to the next question, shown in Figure F.16, illustrate that most of the participants had a favorable view of the idea to have such an application. This indicates that users are open to the notion of using a solution like this and perhaps willing to give it a try, if it can provide some noticeable and measurable benefits compared to traditional alternatives.

Such an application would be useful for finding solutions to programming issues
(6 responses)

Figure F.16: A chart demonstrating the perceived usefulness of having an application dedicated to finding programming-related solutions (such as this web prototype)

The final group of questions focused on the prototype itself. Their aim was to assess which aspects of the application seemed to work well and which did not, so that these deficiencies can be properly addressed in the next iteration of the implementation. One such aspect was the “precision” setting which was intended to refine the output of the prototype and thus help users in their search for a relevant answer. As Figure F.17 shows,
the opinion of the participants on whether that setting had an effect on the results is pretty mixed. Furthermore, views were also rather divided on the exact setting that produced the most useful results. Therefore, it is evident that this functionality, like the whole prototype in general, requires further work before it can meet the standards of quality demanded by the users.

**The "precision" setting had an impact on the output (6 responses)**

![Bar chart showing the perceived impact of the "precision" setting](image1)

**Which "precision" setting provided the most useful results? (6 responses)**

![Pie chart showing the responses](image2)

*Figure F.17: Diagrams illustrating the perceived impact of the "precision" setting of the application as well as indicating which option (if any) produced the most useful results.*

Lastly, the final question of the post-test survey attempted to discover which part of the results were perceived to be most helpful by the people who tested the prototype. The responses shown in [Figure F.18](image3) demonstrate that there is no definitive answer, with a
third of the participants believing that both the solution suggestions and the similar issues were equally helpful, while another third being confident that it was just the similar issues. Nevertheless, since often times users were unable to find what they were looking for, this means that they did not consider any of the output that was presented to them to be helpful, which is the primary concern that must be addressed in the future.

**Which part of the results was most helpful?** (6 responses)

![Pie chart](image)

**Figure F.18:** A chart showing which part of the search results was considered to be most helpful by the study subjects
Appendix G Implementation code

Finally, the last section of the Appendix contains several samples of the code that was written during the process of developing the system. The implementation was done entirely using Python, which, as previously mentioned, is a popular choice in the data science field and is commonly used for many large scale analytical tasks.

**Code sample G.1:** Using the GitHub API to retrieve all project repositories that fit a set of predefined criteria

```python
# Fetch repositories

def get_repos(self, page, stars='>=10000'):
    params = {
        'q': 'language:javascript stars:%s' % stars,
        'sort': 'stars',
        'order': 'desc',
        'per_page': 100,
        'page': page
    }

    url = self.req.build('search/repositories', params)

    return self.req.send(url)
```

**Code sample G.2:** Collecting a sample of issues which meet the requirements of the research, from a given project repository

```python
# Fetch issues from repo

def get_issues(self, repo, page=1):
    if self.rate_limit() > 0:
        self.info['issues_page'] = page
        print('Traversing page %d of issues.' % page)
        print('Collected issues from %s: %d' % (repo, self.info['collected_items']))

    params = {
        'state': 'closed',
        'sort': 'created',
        'direction': 'desc',
        'since': '2015-01-01T00:00:00Z',
        'per_page': 100,
        'page': page
    }

    url = self.req.build('repos/' + repo + '/issues', params)
    res = self.req.send(url)
```

The full code can be found here: [https://drive.google.com/open?id=0BwdNIKiwHxCytNUc05dGtheU0](https://drive.google.com/open?id=0BwdNIKiwHxCytNUc05dGtheU0)
next_page = res["next_page"]
data = res["data"]
self.set_rate_limit(res["calls"])

for issue in data:
    if "pull_request" not in issue:
        num = issue["number"]

        if num > self.info["issue_number"]:
            continue

        self.info["issue_number"] = num

        if self.info["current_index"] in self.info["repo_sample"]:  
            issue_comments = list()

            if issue["comments"] > 0:
                print("Fetching comments for issue \%d (%d)" % ( 
                    self.info["current_index"], num))

                self.get_comments(issue["comments_url"],
                                  issue_comments)

            issue["issue_comments"] = issue_comments

            self.issues.insert_one(issue)

            print("Issue \%d collected." % self.info["current_index"])

            self.info["collected_items"] += 1

            self.info["current_index"] += 1

        # Exit method after gathering the specified total
        if self.info["collected_items"] >= self.info["collect_total"]:
            return

    if next_page is not None:
        self.get_issues(repo, next_page)
    else:
        self.req.wait()
        self.get_issues(repo, page)

---

**Code sample G.3:** Filtering out all non-English issues from the collected dataset

```python
# Remove non English entries
def filter_english(self, data, fields):
    english = set(stopwords.words('english'))
    is_english = dict()
```
Appendix G. Implementation code

column_names = list()

for field in fields:
    is_english[field] = list()
    entries = data[field].str.split(' ')

    for text in entries:
        word_count = 0

        for word in text:
            if word.lower() in english: word_count += 1

        is_english[field].append(word_count > 0)

    column_name = field + '_is_english
    column_names.append(column_name)
    data.loc[:, column_name] = is_english[field]

filtered = data.filter(items=column_names)
has_english = filtered.isin([True]).sum(axis=1) > 0

data = data[has_english]
data = data.drop(column_names, axis=1)

return data

Code sample G.4: An overview of the various text preprocessing activities involved in the preparation of the issue dataset

# Convert text to lowercase
def normalize(self, text):
    return text.lower()

# Split text into separate words
def tokenize(self, text):
    return word_tokenize(text)

# Remove most common English words from text
def remove_stopwords(self, text, stopword_list=list()):
    stop = set(stopword_list) if stopword_list else set(stopwords.words('english'))
    return [word for word in text if word not in stop]

# Remove word suffixes
def stem(self, text):
    stemmer = SnowballStemmer('english')
Appendix G. Implementation code

```python
return [stemmer.stem(word) for word in text]

# Convert words into their root form
def lemmatize(self, text):
    wnl = WordNetLemmatizer()
    ds = self.get_word_pos(text)
    return [wnl.lemmatize(word[0], pos=word[1]) for word in ds]

# Determine POS for each word
def get_word_pos(self, text):
    ds = pos_tag(text)
    return [(word[0], self.get_wordnet_pos(word[1])) for word in ds]

# Transform POS tags from one implementation to another
def get_wordnet_pos(self, treebank_tag):
    return {
        'J': wordnet.ADJ,
        'V': wordnet.VERB,
        'M': wordnet.VERB,
        'N': wordnet.NOUN,
        'R': wordnet.ADV
    }.get(treebank_tag[0], wordnet.NOUN)

# Remove all non-alphabetic chars from text
def strip_alpha(self, text):
    return [word for word in text if word.isalpha()]

# Remove HTML tags from text
def strip_html(self, text):
    return BeautifulSoup(text, 'lxml').get_text()

# Replace chars in text using regex
def strip_chars(self, text, regex, replace=' '):
    return re.sub(regex, replace, text)

# Remove URL's
def strip_url(self, text):
    regex = r'((http|https):/\/)\[A-Za-z0-9_\-\.\+/=\&\%]+'
    return self.strip_chars(text, regex)

# Remove emails
def strip_email(self, text):
    regex = r'\[A-Za-z0-9_\-\.]+@[A-Za-z0-9_.]+\.[A-Za-z]+\{2,\}'
    return self.strip_chars(text, regex)
```

171
Appendix G. Implementation code

# Remove duplicate characters

def strip_duplicate(self, text):
    regex = r'(.)\1\{2,\}'
    replace = r'\1'
    return self.strip_chars(text, regex, replace)

Code sample G.5: Extracting the most relevant terms (i.e. features) from the document dataset

# Extract features from document dataset

def extract_features(self, data, num_features=None, max_freq=0.75, min_count=10, stopwords=True, ngram_range=(1,3),):
    if isinstance(stopwords, list):
        stop = stopwords
    elif stopwords is True:
        stop = 'english'
    else:
        stop = None

    self.vectorizer = TfidfVectorizer(stop_words=stop, ngram_range=ngram_range, max_df=max_freq, min_df=min_count, max_features=num_features, sublinear_tf=True)
    self.features = self.vectorizer.fit_transform(item for item in data)

Code sample G.6: Grouping the text documents into separate clusters, using the K-Means algorithm

# Cluster documents

def cluster(self, features, num_clusters=2, max_iterations=300, logging=False):
    self.model = KMeans(num_clusters, max_iter=max_iterations, verbose=logging)
    self.model.fit(features)
    self.clusters = self.model.labels_

Code sample G.7: Reducing the dimensionality (i.e. the number of dimensions) of the feature vectors

# Reduce data dimensionality

def scale_data(self, features, num_dimensions=2, save_features=False, algorithm='arpack', random_seed=None):
    svd = TruncatedSVD(n_components=num_dimensions, algorithm=algorithm, random_state=random_seed)
    self.scale_func = make_pipeline(svd, Normalizer(copy=False))
    scaled = self.scale_func.fit_transform(features)
if save_features is True:
    self.scaled_features = scaled

return scaled

---

**Code sample G.8:** *Building the hierarchical clustering tree, by dividing the top level clusters into two additional sublevels*

```python
# Build hierarchical cluster tree
def build_hierarchy(self, features, cluster_indices, num_clusters=5, max_cluster_size=100):
    options = {
        'max_level':3,
        'max_cluster_size': max_cluster_size
    }

    self.hierarchy = dict()
    clusters = dict()

    for i in range(len(set(cluster_indices))):
        clusters[i] = list()

    for ind, item in enumerate(cluster_indices):
        clusters[item].append((ind, features[ind]))

    for cluster in clusters:
        self.hierarchy[cluster] = dict()
        options['cluster_index'] = cluster
        options['level'] = 1

        self.subcluster(clusters[cluster], num_clusters, options)
        options.pop('subcluster_index', None)

# Divide clusters into subclusters
def subcluster(self, features, num_clusters, options):
    options['level'] += 1
    subcluster_indices = list()

    model = KMeans(num_clusters)
    f = [item[1] for item in features]
    model.fit(f)

    if 'subcluster_index' in options:
        self.hierarchy[options['cluster_index']][options['subcluster_index']][model] = model
        self.hierarchy[options['cluster_index']][options['subcluster_index']][options['features']] = features
```
else:
    self.hierarchy[options['cluster_index']][model] = model
    self.hierarchy[options['cluster_index']][features] = features

indices = model.labels_
cluster_sizes = Counter(indices)

for index in cluster_sizes:
    if cluster_sizes[index] > options['max_cluster_size']:
        subcluster_indices.append(index)

for index in subcluster_indices:
    if options['level'] >= options['max_level']:
        return

self.hierarchy[options['cluster_index']][index] = dict()
options['subcluster_index'] = index

sub_features = [features[ind] for ind, item in enumerate(indices) if item == index]

self.subcluster(sub_features, num_clusters, options)

options['level'] -= 1

Code sample G.9: Assigning a new issue to the most suitable cluster/s, using the trained categorization model

def predict(self, features, level=1, scale=False):
    if scale is True:
        features = self.scale_func.transform(features)

    if level > 1:
        top_level = self.model.predict(features)
        assignment = list()

        for ind, item in enumerate(top_level):
            item_assignment = list()
            item_assignment.append(item)

            f = features[ind, None, :]
            second_level = self.hierarchy[item][model].predict(f)
            subcluster = second_level[0]
            item_assignment.append(subcluster)

            if level > 2 and subcluster in self.hierarchy[item]:
                third_level = self.hierarchy[item][subcluster][model].predict(f)
item_assignment.append(third_level[0])

assignment.append(item_assignment)

return assignment

return self.model.predict(features)

Code sample G.10: Transforming new issues into vectors, through the use of the existing term vocabulary

```python
# Transform issues into vectors
def vectorize_data(self, data):
    processor = TextPreprocessor()
    processed = processor.prepare(data)

    if not processed:
        return None

    if isinstance(processed[0], list):
        vector = self.clusterer.vectorize([' '.join(item) for item in processed])
    else:
        vector = self.clusterer.vectorize([' '.join(processed)])

    return vector if vector.nnz > 0 else None
```

Code sample G.11: Finding the most similar issues to a given entry, based on the vector distance

```python
# Find similar issues
def find_similar(self, vector, features, issues, max_results=5, max_dist=0.75):
    closest = self.get_closest(vector, features, num_docs=max_results, threshold=max_dist)
    ids = list()

    for item in closest:
        ids.append(issues[item]['id'])

    res = list(self.db.get_issues_in_list(ids, {'_id':0}))

    for i, item in enumerate(res):
        for index, issue_id in enumerate(ids):
            if issue_id == item['id']:
                res[i]['index'] = index

    res.sort(key=lambda x: x['index'])
```
 Agreement G.12: Retrieving the most highly rated comments from the set of most similar issues, identified within the assigned category

```python
# Get closest documents to point
def get_closest(self, vector, features, num_docs=5, threshold=0.75):
    distances = euclidean_distances(vector, features)
    top = distances.argsort()[:, :num_docs]
    indices = [index for index in top[0] if distances[:, index] < threshold]
    return indices
```

```python
# Get most upvoted comments
def get_top_comments(self, issues, num_comments=5, threshold=1):
    comments = [item['issue_comments'] for item in issues]
    all_comments = list(itertools.chain.from_iterable(comments))

    if not all_comments:
        return list()

    cleaner = DataCleaner()
    comm = cleaner.clean(pd.DataFrame(all_comments), ['body'])
    comm = cleaner.filter_english(comm, ['body'])

    comm.loc[:, 'positive_score'] = self.calculate_score(comm['reactions'])
    top = comm.sort_values('positive_score', ascending=False)[:num_comments]
    top = top[top['positive_score'] >= threshold]
    return top.to_dict(orient='records')
```

```python
# Calculate comment score
def calculate_score(self, comments):
    scores = list()

    for comment in comments:
        score = 0

        score += comment['+1']
        score += comment['hooray']
        score += comment['heart']
        score -= comment['-1']
        score -= comment['confused']
        score -= comment['laugh']
```

scores.append(score)

return scores