Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Challenges in React Native Development: A Study of Stack Overflow Posts

Autora: Gabriela Barrozo Guedes
Orientadora: Profa. Dra. Carla Rocha Aguiar

Brasília, DF
2022

Gabriela Barrozo Guedes

Challenges in React Native Development: A Study of Stack Overflow Posts

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Supervisor: Profa. Dra. Carla Rocha Aguiar

Brasília, DF

2022

Gabriela Barrozo Guedes
Challenges in React Native Development: A Study of Stack Overflow Posts/ Gabriela Barrozo Guedes. – Brasília, DF, 2022-
63 p. : il. (algumas color.) ; 30 cm.

Supervisor: Profa. Dra. Carla Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2022.

1. React Native. 2. Mobile development. I. Profa. Dra. Carla Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Challenges in React Native Development: A Study of Stack Overflow Posts

CDU 02:141:005.6

Gabriela Barrozo Guedes

Challenges in React Native Development: A Study of Stack Overflow Posts

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 29 de junho de 2022:

Profa. Dra. Carla Rocha Aguiar
Orientador

Prof. Dr. Renato Coral Sampaio
Convidado 1

Profa. Dra. Milene Serrano
Convidado 2

Brasília, DF
2022

Abstract

Mobile App development is growing with smartphones and the significant advances in technologies. Therefore there is more use of mobile app development tools to create those apps.

React Native is one of the most popular frameworks for hybrid development. This framework makes it possible for the developer to write one code in JavaScript and generate apps for iOS and Android. With the rise in the development of mobile apps, more developers are using React Native, creating a community that helps each other with developing issues.

This study aims to analyze the community of React Native developers by investigating the difficulties reported on Stack Overflow. This study used Scoccia, Migliarini e Autili (2021) as a base project for the dataset analysis with the Mallet tool and improved it to be reused by other datasets.

Key-words: react-native, development, mobile app, stack-overflow, mallet.

Resumo

O desenvolvimento de aplicativos móveis está em expansão devido ao crescente uso de smartphones e aos avanços significativos da tecnologia, de forma que o uso das ferramentas de desenvolvimento para criar esses aplicativos também aumentou.

React Native é um dos frameworks mais populares para desenvolvimento híbrido. Esse framework possibilita que o desenvolvedor escreva um código em JavaScript e obtenha aplicativos para ambos os sistemas Android e iOS. Com o crescimento de aplicações móveis, mais desenvolvedores estão utilizando React Native, criando assim uma grande comunidade que se ajuda com dúvidas de desenvolvimento.

Esse estudo tem como objetivo analisar a comunidade de desenvolvedores de React Native através das perguntas feitas pelo Stack Overflow. Utiliza como base para o desenvolvimento, o projeto de Scoccia, Migliarini e Autili (2021) que possui scripts para a utilização do Mallet que são evoluídos para se adaptar a datasets diferentes.

Palavras-chave: react-native, desenvolvimento, aplicações móveis, stack-overflow, mallet.

List of Figures

List of Tables

List of abbreviations and acronyms

RAM	Random-access memory
GPS	Global Positioning System
OS	Operating System
API	Application Programming Interface
SDK	Software Development Kit
LDA	Latent Dirichlet Allocation
Q&A	Questions and Answers
SQL	Structured Query Language
CSV	Comma Separated Values
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

Contents

1 Introduction

Software technology is expanding to automate processes that used to be made manually or in person. An example is food delivery, stores, etc. The use of smartphones and other mobile devices is also growing significantly. More people opt only to have a smartphone instead of having a computer. The higher use of mobile devices and the automated process results in the growth of mobile apps and the use of mobile apps development tools to create those apps.

One of those tools is React Native (2021b), an open-source framework for mobile development that builds iOS and Android apps. React Native was created in 2015 by Facebook. It is a highly used framework with an extensive community.

Developers communities commonly use forums to communicate, interact and help each other in solving issues. Stack Overflow (2021b) is a questions and answers website that is highly used in those developers' communities, especially in solving problems that come with technology. React Native community also makes use of Stack Overflow, having over 107.468 questions under the react-native tag.

This study proposes the analysis of the React Native questions on Stack Overflow to identify the issues and difficulties of using this framework. The analysis of the main topics under the React Native community will help understand this community better and visualize what aspects the framework needs to evolve to attend to its users. This work is based on an open-source repository and aims to structure the original repository to allow other people to create similar studies using this method.

1.1 Objectives

This study aims to evolve the Scoccia, Migliarini e Autili (2021) project to analyze the React Native community and identify developers' main challenges when using this framework. The following goals must be accomplished to achieve the study's initial goal.

- Fetch the questions under the react-native tag on Stack Overflow
- Evolve the Scoccia, Migliarini e Autili (2021) scripts to use other datasets
- Divide the question into topics with the scripts
- Analyze the results

1.2 Work Structure

This work presents itself with a background chapter, which explains the context of meaningful topics addressed in the study and related works. A proposal chapter that defines the study's proposal, research questions, methodology, and technologies used. A results chapter shows the dataset first analysis, experiments made to understand the project scripts, and the processes to evolve the project repository. And finally, a chapter for the study's conclusion.

2 Background

2.1 Mobile App Development

Mobile app development is growing with the rise of technologies and mobile devices. Unlike web apps where the software providers are responsible for selecting the machine where the service will run, an app on a mobile phone has limited resources, such as storage memory, RAM, and battery, and the users are responsible for the device selection. The software may run on an incredible amount of different phones, dealing with different types of resources. Despite having similar features to the web app, it can access many more functionalities, such as GPS, camera, Bluetooth, etc.

Besides the resources, a mobile phone also may have different types of operating systems. Nowadays, as shown in Figure 1, there are two leading OS, iOS and Android, each one of them with different official development approaches and tools. An Android app uses Android Studio to develop and build apps written on Java or Kotlin. Therefore an iOS app has XCode as a developing tool and is written in Objective-C or Swift.

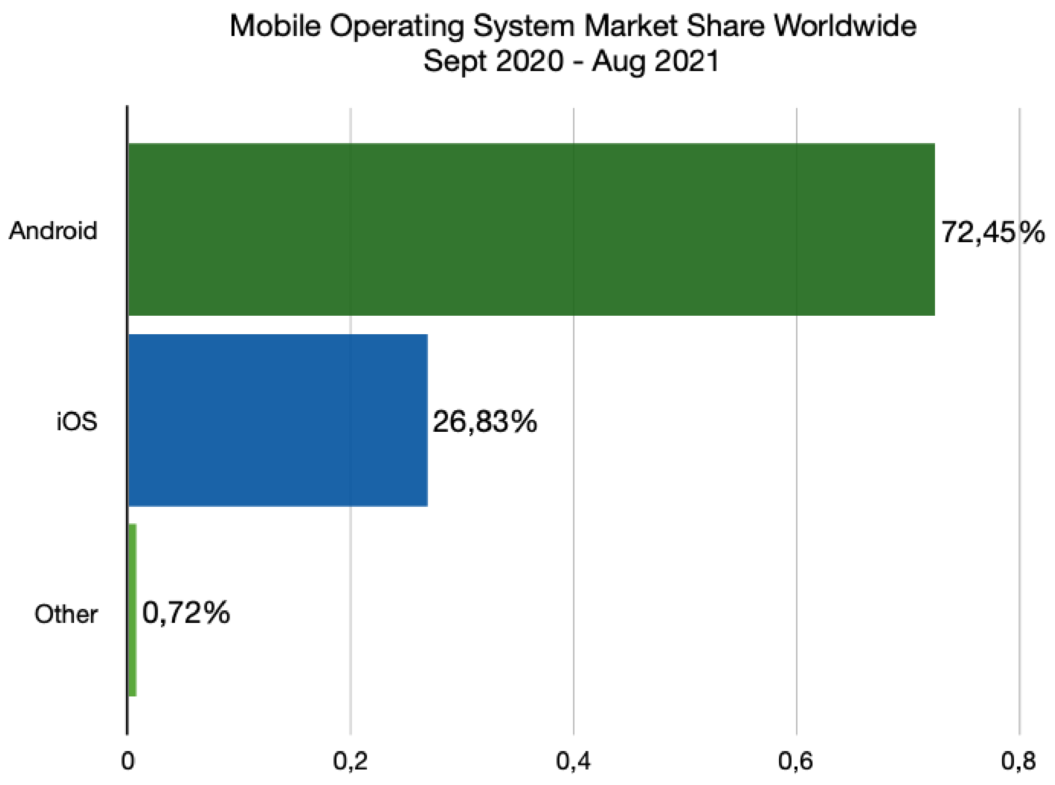


Figure 1 – Mobile OS Chart (StatCounter, 2021)

Through time, many mobile app development frameworks emerged to help the developer accelerate the development cycle. Besides the official development supported by each OS company, there are many ways to develop a mobile app. Nunkesser (2018) suggests the division of three different main types of development: Pandemic, Endemic and Ecdemic, as depicted in Figure 2. An Endemic App is made in a traditional development supported by the OS vendors. A Pandemic App is an app made using tools supported by most mobile OSs, for example, a web app made with HTML and CSS or a C/C++ game. As for an Ecdemic App uses cross-platform frameworks with a not endemic language, such as Xamarin.

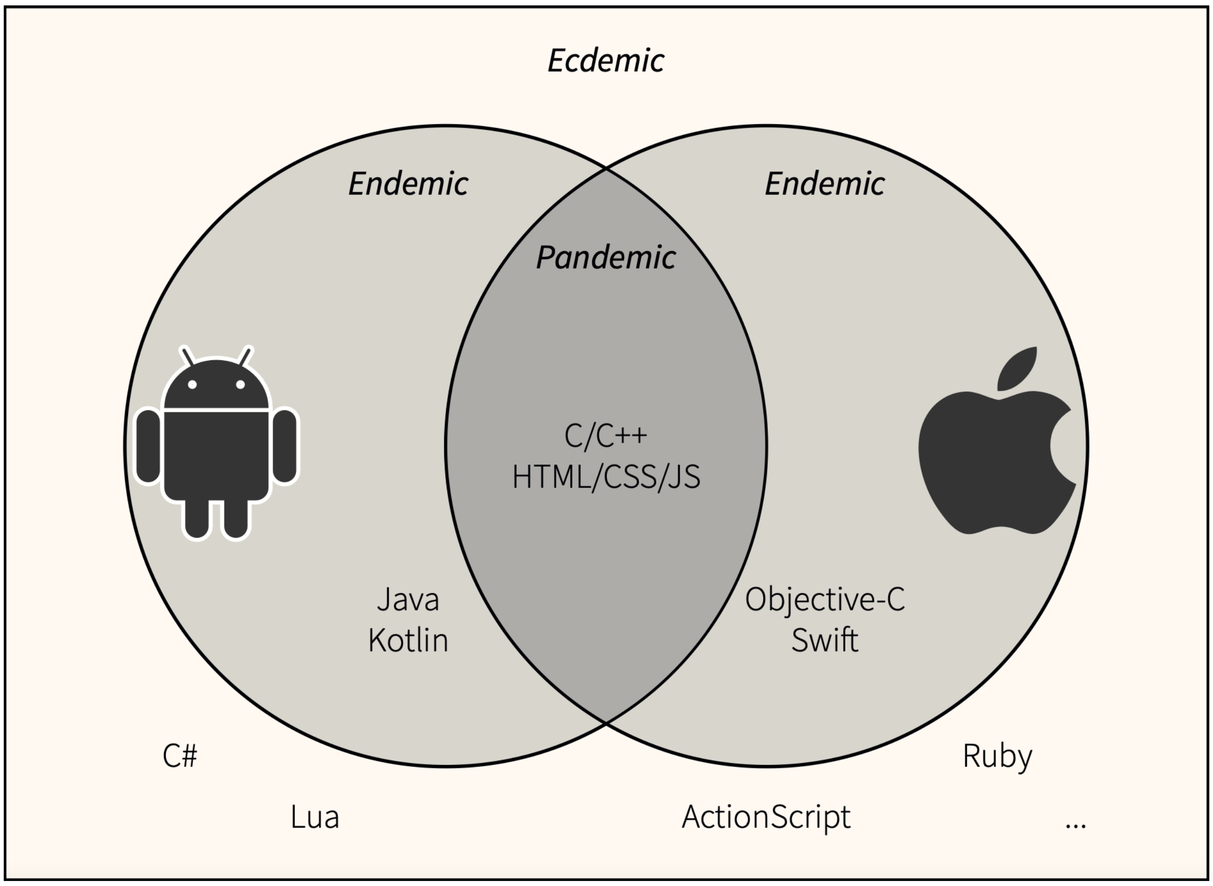


Figure 2 – Mobile Development (NUNKESSER, 2018)

As for a more traditional taxonomy, Smutný (2012) defines Native and Hybrid Apps. A Native App is an app developed in the device's native runtime environment, which has access to all the device's facilities. It is, however, tied to the device's operating system, as for apps built in Java or Kotlin for Android and Objective C or Swift for iOS. Hybrid Apps, otherwise, rely on frameworks that ensure cross-platform compatibility and provide access to the device's facilities.

2.2 Hybrid Solutions

Hybrid Apps, such as Native Apps, are installed on the device and have access to the device's facilities only in a more limited way. Unlike the native approach, this approach allows web app development tools to target multiple mobile platforms (MEIRELLES et al., 2019).

Frameworks for Hybrid Apps uses a combination of web app-development tools, such as HTML, CSS, and JavaScript, and wrappers that contain a cross-platform API that bridges all the requests from the web-based code to the corresponding Platform's API. These frameworks also contain a native wrapper that allows the apps to be distributed on multiple app stores. This type of development solves the portability problem of the native apps. By enabling the development of only one code source that generates an app distributed on multiple platforms, with minimal changes. (MALAVOLTA et al., 2015)

2.3 Differences Between Native and Hybrid Development

Native and Hybrid development are two approaches that must be evaluated before starting a mobile app project. The difference between Native and Hybrid is a layer that translates the request to the platform's API on the hybrid development, which is evident when looking at Figures 3 and 4.

2.3.1 Native Development

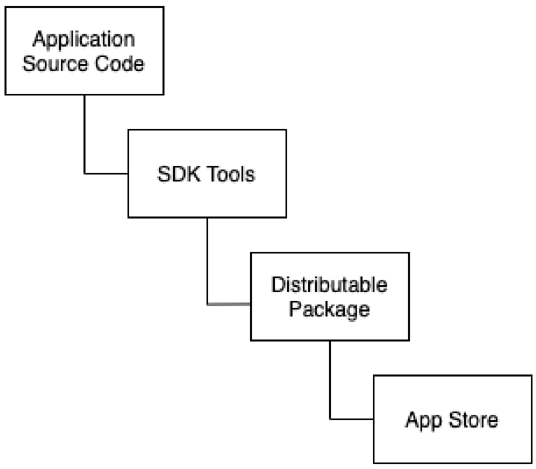


Figure 3 – Native Mobile App (TUN, 2014)

Native development has direct access to SDK Tools, which gives higher performance and direct access to the device's facilities and integrations. It also has the advantage of being up to date with the device's technology since the same company maintains it. However, it only works for one type of operating system, limiting its number of users. To deal with this problem it is necessary to build another app for another OS, increasing cost and development time. (TUN, 2014)

2.3.2 Hybrid Development

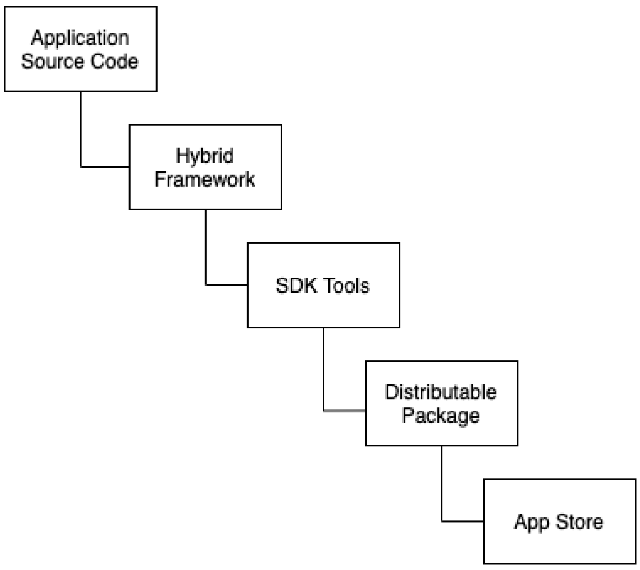


Figure 4 – Hybrid Mobile App (TUN, 2014)

Hybrid development reaches more users since it targets more than one OS, making it a cheaper and faster alternative. Nevertheless, its additional layer provides less support to the device's integrations and impacts its performance since it is one more layer to process (TUN, 2014).

2.4 React Native

React Native (2021b) is a JavaScript framework for hybrid mobile solutions created by Facebook in 2015. React Native is one of the most popular frameworks for hybrid mobile app development. In 2018 it had the second-highest number of contributors for any GitHub repository according to Octoverse (2019). Currently, 27.509 public repositories match the React Native topic on GitHub (2021), and the React Native package on npm (2021) has 486.189 weekly downloads.

React Native uses the React JS framework and native iOS and Android components. Using this framework, the developer writes one code in JavaScript and builds the app on Android and iOS. As the React Native (2021a) documentation explains, the framework is built on top of native components. Therefore, it renders as the operating system design. In a native Android app, a view is developed in Java or Kotlin. As in iOS, it is developed in Swift or Objective-C. In React Native, these views are created at runtime for their corresponding system. React Native also allows the developer to build native components for Android or iOS and use them on the app when the hybrid JavaScript code does not meet the app's needs.

This framework is popular in the mobile development community for multiple reasons: using JavaScript, one of the most used programming languages; Facebook's constant support and improvement on the framework; high performance; being a free, open-source framework with an active community supported by Facebook; building the app on iOS and Android, making the development cheaper; possibility to combine Hybrid development with Native components; (ZOHUD; ZEIN, 2021)

2.5 Stack Overflow

Stack Overflow is a questions and answers website that focuses on developers' technical difficulties. When Stack Overflow user encounters a problem they do not know how to solve alone, they publish a question. Other Stack Overflow users, seeing the question with a problem they know how to solve, answer it and help the other. The answered questions are kept open for anyone with the same difficulty to see and solve their problems faster.

Stack Overflow (2021b) is one of the 50 most popular websites globally, receiving over 100 million visitors per month, and has more than 21 million questions registered. This website is also highly used for react-native developers, having 107.468 questions under the react-native topic, making Stack Overflow an excellent database for storing most of the difficulties of using this technology.

2.6 The Developer's Profile

Stack Overflow periodically researches the developer persona, which is their website's primary user and the authors of the questions posted. Analyzing the 2021 survey (Stack Overflow, 2021a), we can conclude that most developers are from the United States of America (18,33%). Also that 32,52% are between 25 to 34 years old, and 91,67% are men. On a race and ethnicity aspect, 58,42% are white or European descendants.

When looking at their experience, we see that 59,53% learned to code from online resources. Most developers have 5 to 9 years of experience (29,91%), while developers with less than five years represent only 19,62%. As for education, most have a bachelor's degree (42,37%).

When analyzing the type of developer, we see that mobile developers represent 14,74% (the 5th position). Seeing the React Native statistics, we see that JavaScript (the framework's primary language) is the most popular programming language, with 64,96%. React JS (the framework on which React Native is based) is the most used web framework with 40,14%. React Native is the 6th non-web framework most used, with 14,51%. On the loved or dreaded analysis, 58,08% love React Native.

2.7 Open source software

Open source is a initiative that comes from the Free Software concept. Free software is a software developed under a license that guarantees users the rights to run, copy, distribute, study and change the software (GNU, 2022). The "free" term is considered as freedom, not as free of price (DIBONA; OCKMAN, 1999). Free Software characterizes itself by the four essential freedoms as described in DiBona e Ockman (1999):

- The freedom to run the program;
- The freedom to change the program by accessing the source code;
- The freedom to copy and distribute the program;
- The freedom to redistribute modified copies;

Developing open source software has been adopted globally and proved a valuable approach. This type of development can be a catalyst for better practices of development. It also creates software communities and is an excellent way to disseminate software innovations and research. (FUGGETTA, 2003)

2.8 Topic Modeling and The Mallet Toolkit

Mallet (Machine Learning for Language Toolkit) is a Java package for natural language processing. For document classification, clusterization, topic modeling, or other machine learning application on text. (MCCALLUM, 2002)

The Mallet tool applies the Latent Dirichlet Allocation (LDA) algorithm, a popular topic modeling algorithm created by Blei, Ng e Edu (2003). Topic modeling tools such as the LDA algorithm look for patterns in the use of words, attempting to identify the document's semantics. Topic models extract topics from a particular text by assuming each word in a text is from a particular topic, making it possible to decompose a text and identify the topics that compose it. (GRAHAM; WEINGART; MILLIGAN, 2012)

Some words harm the topic modeling assumption that every word is from a specific context. The tool categorizes topics based on words, and some words may often be enough for the tool to categorize it as a topic without having the context to make a topic, misleading the algorithm. Those misleading words are referred to as stopwords, which are removed from the dataset during a pre-processing step. Some examples of stopwords are "each", "about", "such", "the" and "and". (SARICA; LUO, 2021)

The Mallet tool is applied in many kinds of research related to identifying document contexts. For example, Novel Approach to Cluster Patient-Generated Data Into Actionable Topics: Case Study of a Web-Based Breast Cancer Forum (JONES et al., 2018) researches a breast cancer forum using the mallet algorithm. Another example is Systematic Mapping Study on Software Engineering for Sustainability (SE4S) (PENZENSTADLER et al., 2014) which uses Mallet to study the topics being discussed under the Software Engineering for Sustainability context.

2.9 Related Works

Similar to the purpose of this paper, Challenges in Developing Desktop Web Apps: a Study of Stack Overflow and GitHub (SCOCCIA; MIGLIARINI; AUTILI, 2021) works is a study that researches topics under web apps development. This research used the Mallet tool to filter StackOverflow and GitHub Issues topics. They searched for questions over NW.js and Electron frameworks to find relevant topics on developing these frameworks and evaluate which ones are more difficult for the developers.

Challenges in Chatbot Development: A Study of Stack Overflow Posts (ABDELLATIF et al., 2020) studies topics under chatbot development. This work inspired the Scoccia, Migliarini e Autili (2021) research and also uses the LDA algorithm with Mallet to find StackOverflow topics, which in this case are for chatbot development. This study researches the difficulties of chatbot developers by analyzing what those developers ask questions and which ones are most difficult to answer.

3 Proposal

This work is based on the methodology presented in Scoccia, Migliarini e Autili (2021). They study the Stack Overflow and GitHub data searching to identify the difficulties of developing a desktop web app. This work will use the same data collection process and analysis of Stack Overflow questions under the React Native context, adapting their repository to be used in different contexts.

3.1 Research Questions

This study aims to analyze the React Native community, map the main difficulties in developing React Native apps, and understand the developer using this framework. This work will allow the community to accurately attack the framework's problems by acknowledging the user's pain (the React Native Developer). This research is done by using the Scoccia, Migliarini e Autili (2021) methodology and adapting their repository to receive and run different types of research, creating a software community around it. This study aims to answer the following research questions.

RQ. 1 *What are the topics on React Native development developers ask questions?*

RQ. 2 *How to evolve the Scoccia, Migliarini e Autili (2021) project to analyze different stack overflow datasets?*

Using the LDA algorithm, this study will collect the Stack Overflow questions and cluster them into more specific topics under the react-native context. The list of topics gathered from this process will answer **RQ. 1**.

This study will involve a contribution to Scoccia, Migliarini e Autili (2021) project where the project will be structured to receive different inputs to be used in different researches. **RQ. 2**.

3.2 Methodology

The diagram in Figure 5 was made to illustrate the methodology used and how the steps are used to answer the Research Questions, showing all the steps to be followed in this study.

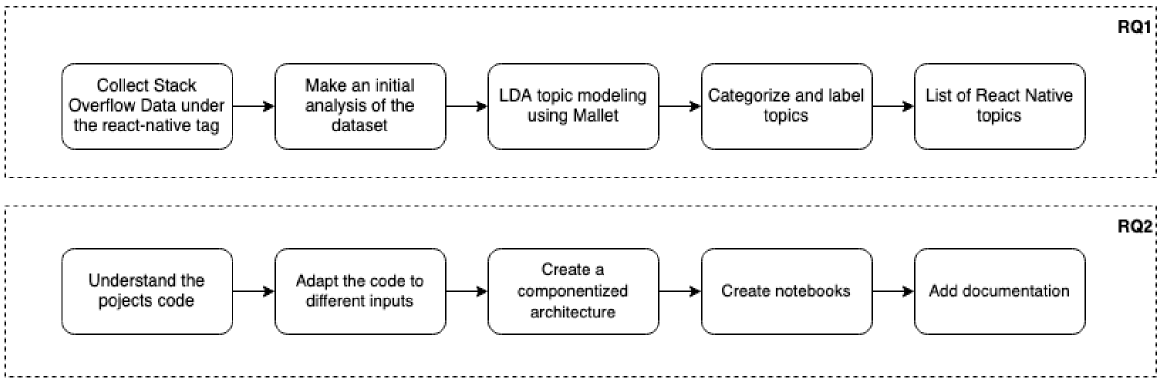

RQ1
Collect Stack Overflow Data under the react-native tag
Make an initial analysis of the dataset
LDA topic modeling using Mallet
Categorize and label topics
List of React Native topics
RQ2
Understand the pojects code
Adapt the code to different inputs
Create a componentized architecture
Create notebooks
Add documentation

Figure 5 – Methodology Diagram

3.3 Data collection

The first step is to collect technical discussions about React Native. For this purpose, the Stack Overflow website is an excellent target since it stores many discussions on this topic. The Stack Exchange Data Explorer (2021) will be used to collect the data. It allows us to compose a SQL Query to search the Stack Overflow database.

The following queries will be used to search questions with the react-native tag. The Stack Exchange Data has a limit of 50.000 rows. Therefore, it will not contain all the Stack Overflow questions on the react-native topic. The script will need to run several times to collect the complete list of discussions around React Native. The script orders its result by Id. The Id is compared on the second and third run, so the following query continuous the one before.

```
SELECT *
  FROM Posts as Questions
  WHERE
    Questions.PostTypeId = 1
    AND Questions.Tags LIKE '%<react-native>%'
  ORDER BY Questions.Id DESC
```

Listing 1 – First query to retrieve Stack Overflow data

On the query below, the Id comparison number will be changed to equal the last Id number in the query's result table before.

```
SELECT *
  FROM Posts as Questions
  WHERE
    Questions.PostTypeId = 1
    AND Questions.Id < 40603514
```

```sql
   AND Questions.Tags LIKE '%<react-native>%'
ORDER BY Questions.Id DESC
```

Listing 2 – Following query to retrieve the remaining data

The queries return a table that is possible to download in a CSV format. The tables will be merged into one dataset. The result table will contain all the questions stored data, which have the attributes depicted in Table 1.

Questions' attributes
Id
PostTypeId
AcceptedAnswerId
ParentId
CreationDate
DeletionDate
Score
ViewCount
Body
OwnerUserId
OwnerDisplayName
LastEditorUserId
LastEditorDisplayName
LastEditDate
LastActivityDate
Title
Tags
AnswerCount
CommentCount
FavoriteCount
ClosedDate
CommunityOwnedDate
ContentLicense

Table 1 – Data returned from the query in Stack Exchange.

As the database contains many questions, after the topics are divided, this analysis will use the first five questions of each topic to identify its context.

3.4 Data processing

This step aims to analyze the initial dataset and group discussions into specific contexts to understand the main topics addressed. Python (2021) and Jupyter (2021) Notebooks will be used to make a preliminary analysis and evaluate if the data can be used as expected. The notebook will use the initial dataset to collect metrics to validate the dataset's quality.

An automated algorithm will cluster the discussions into related topics, and quality analysis will validate the topics created. The Latent Dirichlet Allocation (LDA) (BLEI; NG; EDU, 2003) algorithm analyzes a document and identifies which group of pre-designated topics compose it. This algorithm assumes that documents under the same topics use a similar group of words, that every document contains many topics, and that every topic is composed of a distribution of words. The LDA algorithm can also work backward and identify a number of topics, given a set of documents.

With the initial dataset, the Mallet tool (MCCALLUM, 2002) will be used. It is a software that uses the LDA algorithm and separates the content into related topics, making it possible to cluster the questions on the dataset on a higher level of abstraction. The algorithm needs an entry parameter K that will be the number of generated topics. K will be set by experimenting with a range of numbers to choose from, evaluating if the generated topics make sense. This process will be rerun after the best range is specified, incrementing on each turn until finding the best outcome.

This process will adapt Scoccia, Migliarini e Autili (2021), which is available as an open-source project on https://github.com/gianlucascoccia/MSR2021Replication.

3.5 Data Analysis

This step analyzes the generated clusters to extract each group's typical characteristics to name, understand, and qualify the topic. It also aims to map the most relevant topics and answer the research questions.

After the automated division of clusters, it is necessary to understand what makes each cluster a topic identified by the LDA algorithm. Each cluster will be analyzed by reading the set of posts and summarizing what they have in common.

3.6 Evolve the repository

This study will use Scoccia, Migliarini e Autili (2021) work repository to follow their methodology and steps to separate the dataset into different topics with the Mallet tool (MCCALLUM, 2002). Their repository consists of helper codes to structure the Mallet outputs and clean the inputs from their original dataset. This study will evolve their work to be used in different contexts. The code must be able to receive inputs other than the original dataset used in Scoccia, Migliarini e Autili (2021) work. This contribution process will follow the steps bellow:

1. Understand the code

The first step to using and contributing to the repository is to understand the code and how to run it. Get the original dataset and run the scripts, analyzing what each script does.

2. Adapt the code to different inputs

In this step, it is necessary to remove the hardcoded paths to the original dataset and adapt the code to receive different datasets and process them.

3. Create a componentized architecture

Componentize the code into modules, so the functions can be called more straightforwardly, in which the users do not need to know all the processes made.

4. Create notebooks

The original code does not have much documentation on how to run the scripts or what order they should be run. Part of the contribution is to create a Jupyter Notebook in which all steps will be explained and ready to be executed in the correct order to run the Mallet tool correctly and fetch the results.

5. Documentation

Add documents necessary for good open-source software practices.

3.7 Technologies

In this section we describe the set of tools, technologies adopted to develop the research.

3.7.1 Git

Git (2021) is an open-source system for version control created by Linus Torvalds to help developers track code versions and control changes made, being able to undo them. All code developed in this study will be stored on a Git repository.

3.7.2 Stack Exchange

Stack Exchange (2021) is a network with over 173 Q&A communities, including Stack Overflow. Stack Exchange also provides a Data Explorer where it allows query searches on the communities database. In this study, this tool will be used to search react-native questions on Stack Overflow.

3.7.3 SQL

SQL (Structured Query Language) is a language used to search and manipulate databases. The Stack Exchange Data Explorer uses SQL to formulate queries. A SQL script will be used to make the query to search Stack Overflow questions on the react-native topic.

3.7.4 Mallet

Mallet (MCCALLUM, 2002) is a tool to process natural language and cluster documents into specific topics. It uses the LDA algorithm to make the topic modeling, which will be used in this study to identify more specific topics in the react-native discussions.

3.7.5 Python

Python (2021) is an open-source programming language with an active community and multiple libraries. Python will be used in the study to manipulate the collected data and gather metrics.

3.7.6 Jupyter

Jupyter (2021) notebook is an open-software web application to make documents with live Python code cells. The notebook is used to make the preliminary analysis of the dataset and in the contribution to simplify the use of the scripts.

4 Results

4.1 Exploratory phase

The initial dataset was built using the Stack Exchange Data Explorer and the queries described in the methodology's data collection step. The questions' Ids ordered the queries. The first query returned a table with 50.000 rows, of which the last row had the Id equal to 58759902. The second query searched for questions with the Id under 58759902. It returned 50.000 rows, and the last row had the Id equal to 40603514. The third and last query searched for questions with an Id less than 40603514. The result was a table with 7.468 rows.

The three queries results were downloaded as a CSV files. The three files were merged to build one dataset with all Stack Overflow questions under the react-native tag, which is the dataset for this study. The generated table has 107.468 rows and the expected columns for the questions' attributes.

The initial analysis was made using Python and Jupyter notebook, with the help of Pandas (2021), Word Cloud (MUELLER, 2020) and D-Tale (Man Group, 2021) libraries to explore the table and get metrics from the dataset. All the work made is stored in a GitHub repository on https://github.com/gabibguedes/TCC_preliminary_results.

4.1.1 Word Clouds

Using the Word Clouds library, the dataset generated word clouds for the questions' Title, Body, and Tags to make an initial verification if there are different subjects to be identified and clustered. An array of words was built to filter the words shown on the clouds to make them more straightforward. The words placed on the array were conjunctions and other common words that would have too much impact on the word cloud but were not relevant for the analysis. The questions' Body is built with HTML and has a lot of standard tags. Those tags were added to the filter array to clear the body's word cloud.

The function below was used to build those word clouds by manipulating the Pandas' table to join the attributes from the column to be analyzed and remove unnecessary words.

```python
def make_wordcloud(column_name):
    column = df.dropna(subset=[column_name], axis=0)[column_name]
    all_inputs = ' '.join(s for s in column)

    stopwords=['and', 'to', 'how', 'is', 'in', 'on', 'with',
               'p', 'i', 'href', 'gt', 'lt', 'quot', 'pre',
               'code', 'nofollow', 'noreferrer', 'style', 'png',
               'rel', 'com', 'http', 'amp', 'imgur', 'blockquote',
               'this', 'github', 'https', 'www', 'alt', 'li', 'a',
               'of', 'the', 'from', 'here'
               ]

    wordcloud = WordCloud(stopwords=stopwords,
                          background_color='white',
                          width=1600, height=800).generate(all_inputs)

    wordcloud.to_file('../images/questions_{}_wordcloud.png'
                          .format(column_name.lower()))
```

Listing 3 – Function to create word clouds

On the Title's (Figure 6) and the Tags' (Figure 7) word clouds, words of different topics are evident. A few examples found are: *Navigation*; *Components*; *Flatlist*; *Android*; *iOS*; *Expo*; *TypeScript*; *JavaScript*; *Firebase*; *Redux*; *Database*; etc.

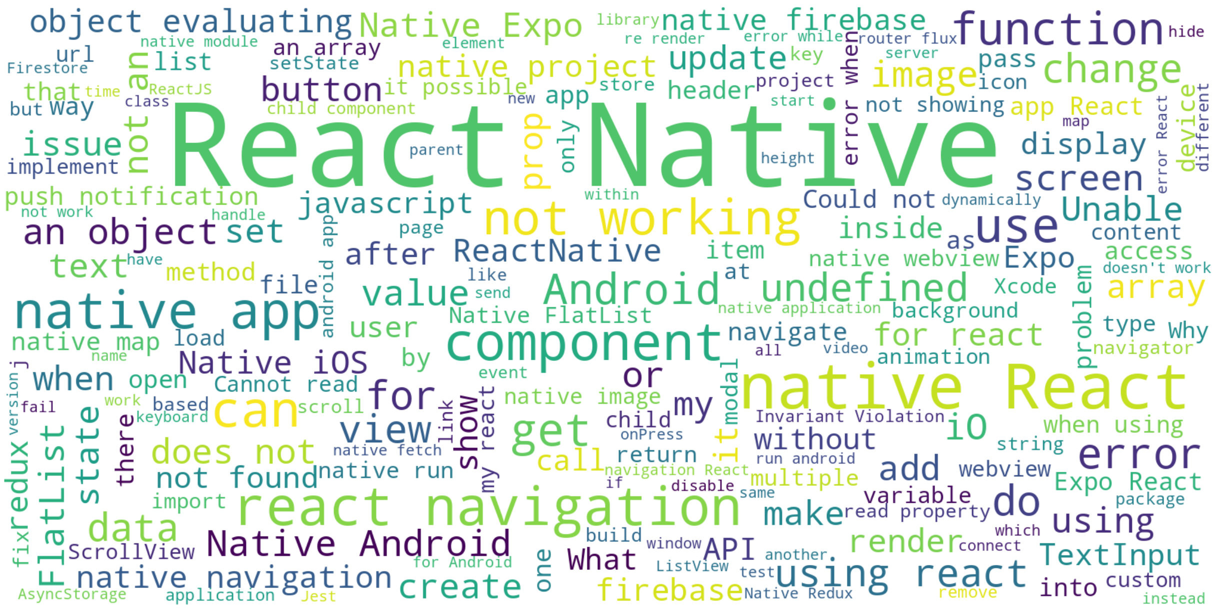


Figure 6 – Questions' Title Word Cloud

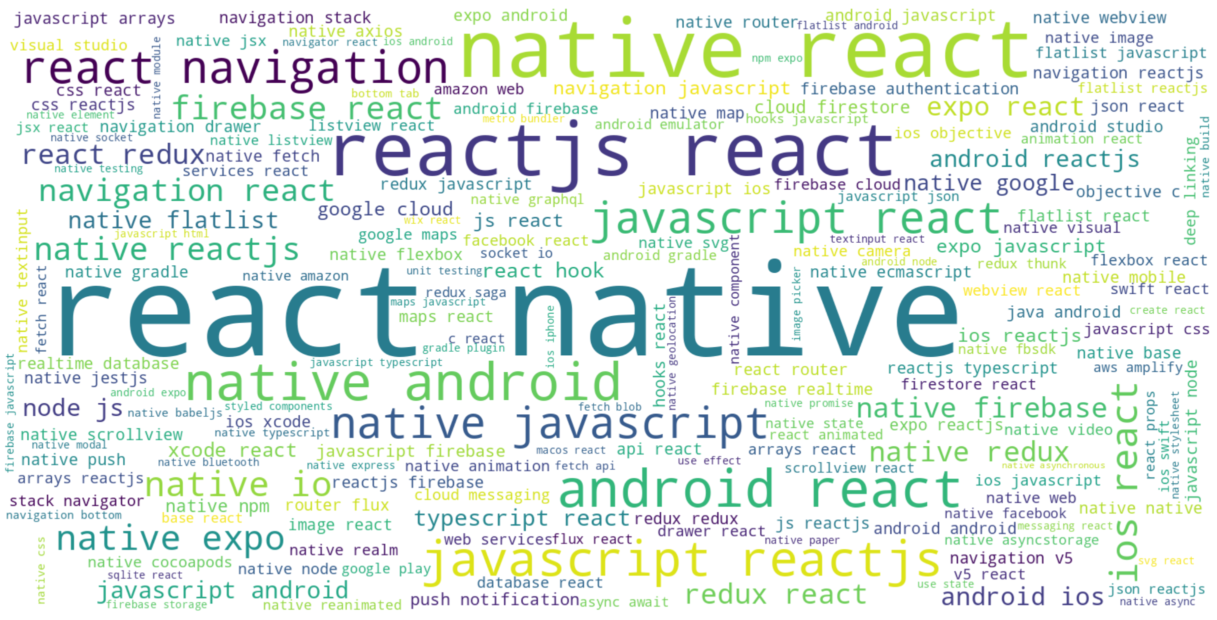


Figure 7 – Questions' Tags Word Cloud

The Body's word cloud in Figure 8 is not as straightforward as the others. As for the type of input, which uses more words and some code snippets to explain the problem. There are a lot of common words that cannot be placed in one specific context, as *Console log*; *Return*; *Have*; *Text*; *Export*; *Default*; etc. However, there can still be identified words to specific contexts as: *TouchableOpacity*; *StyleSheet*; *Navigation*; *setState*; *node_modules*; etc.

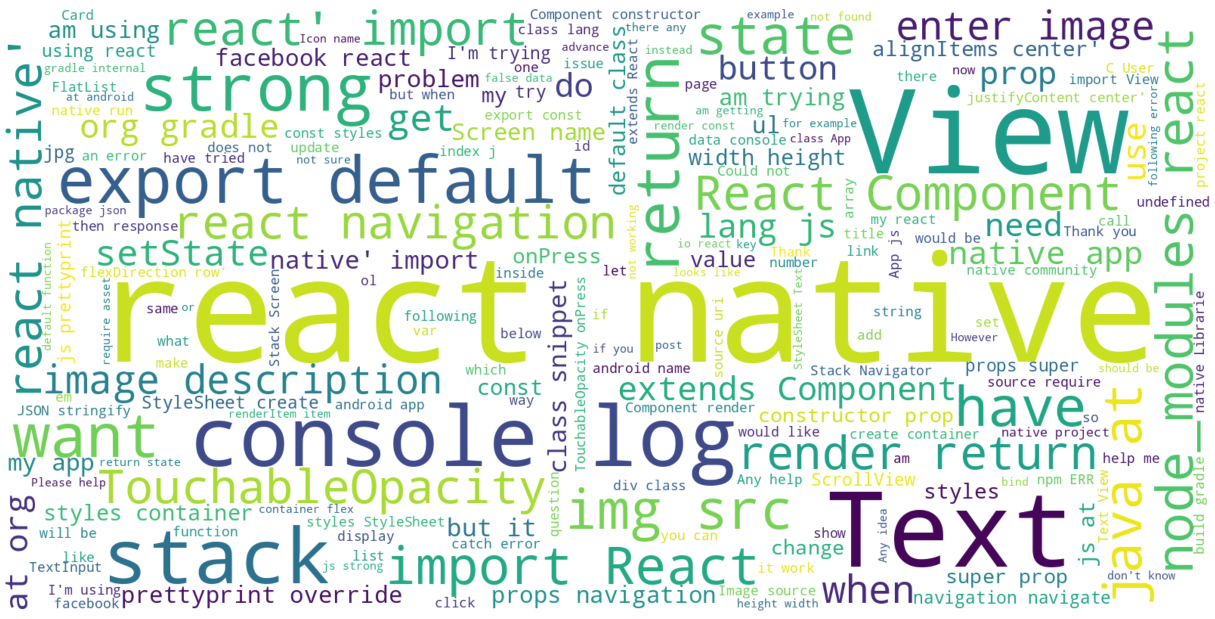


Figure 8 – Questions' Body Word Cloud

4.1.2 Data analysis over time

The D-Tale library was used to generate graphs of the questions' attributes' values over time. Using the CreationDate value, modified only to contain the day information, removing the hours' data. The result value placed the questions on a timeframe of their creation day.

The first generated graph is the number of created questions over time (Figure 9), in which we can see the number of questions under the react-native tag grew as the framework became more used and popular.

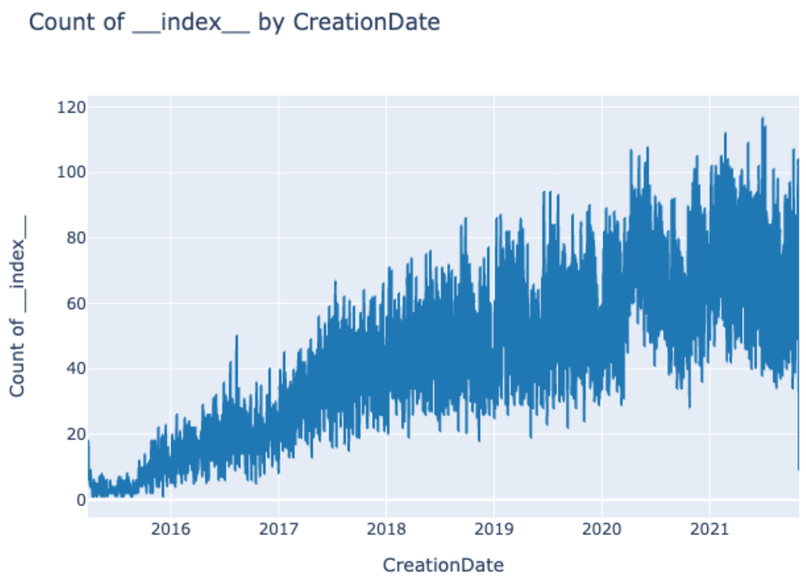


Figure 9 – Amount of questions by time

The questions' attributes show different graphs. A clear example of how time influences the questions' attributes is the Comments (Figure 10) and Answers (Figure 11) count. It mostly followed the technology's growth. However, a question still needs time to collect a considerable amount of comments and answers. That is why the number of most recent comments and answers fall.

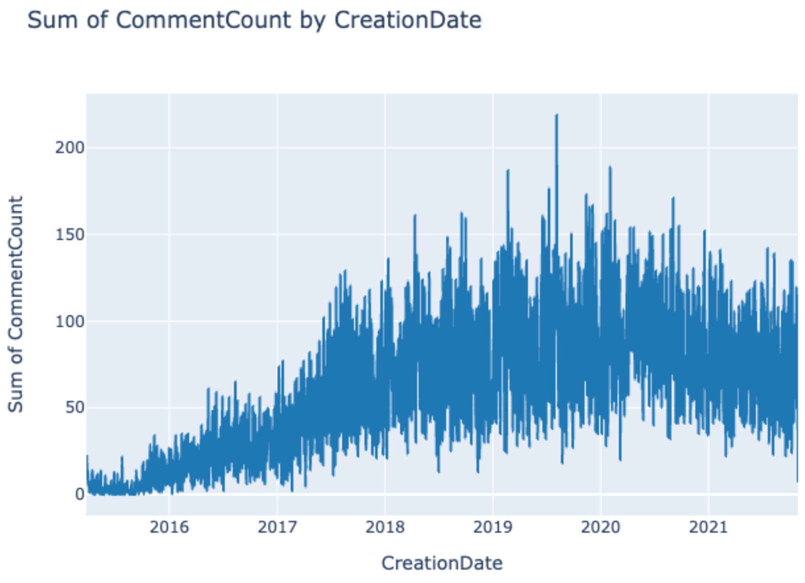


Figure 10 – Amount of comments by time

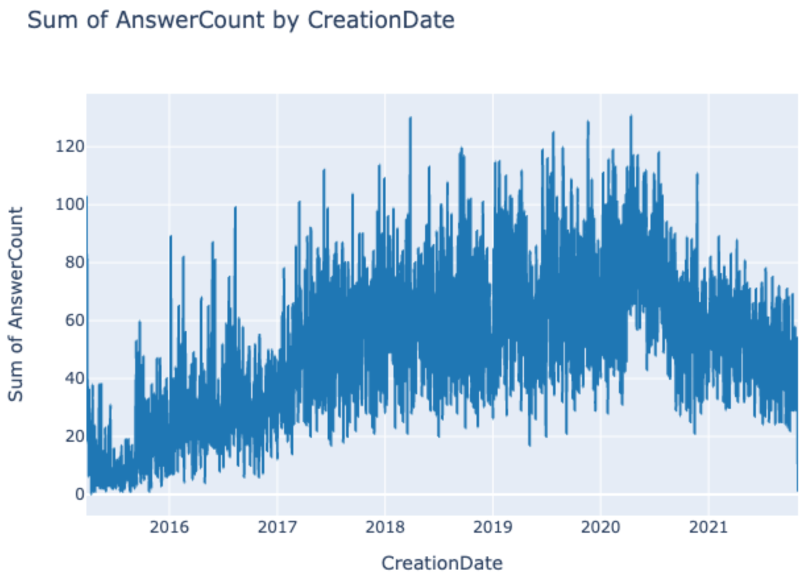


Figure 11 – Amount of answers by time

The Score (Figure 12), Favorite (Figure 13) and View (Figure 14) graphs are influenced by time. The older the post, the more interactions it can accumulate, however, it still seems to have an average value that dictates over time, and only a few posts seem overly popular.

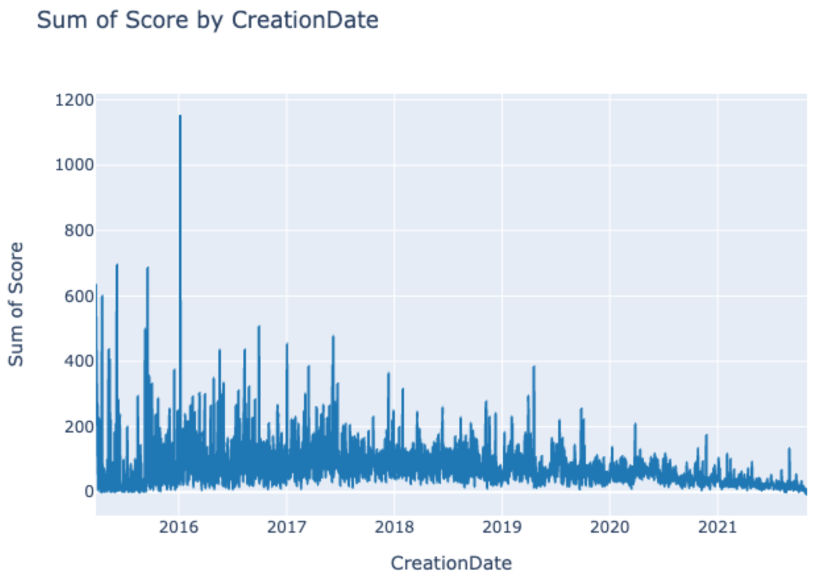


Figure 12 – Amount of score by time

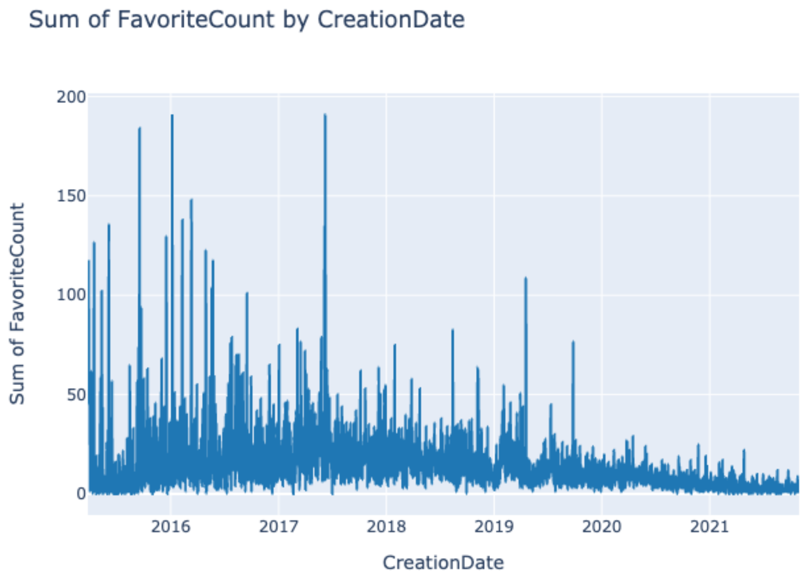


Figure 13 – Amount of favorite by time

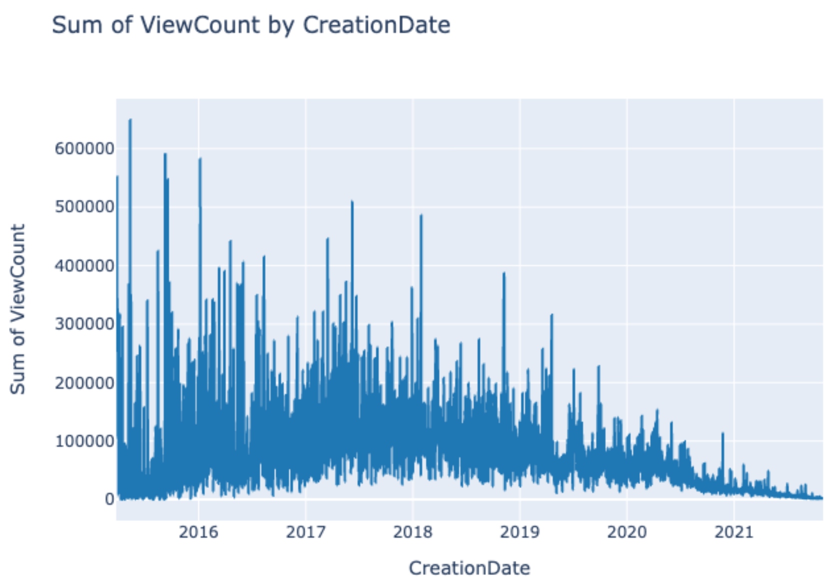


Figure 14 – Amount of views by time

4.1.3 Value analysis

D-Tale library was also used to analyze the values of those attributes. As can be seen on the tables below, the Answer Count(Figure 15), the Comment Count (Figure 16), and Score (Figure 17) usually value smaller than 10. The Favorite Count (Figure 18) reaches values a little higher but still mostly smaller than 20. The attribute that uses more significant values is the View Count (Figure 19) that reaches hundreds of visualizations.

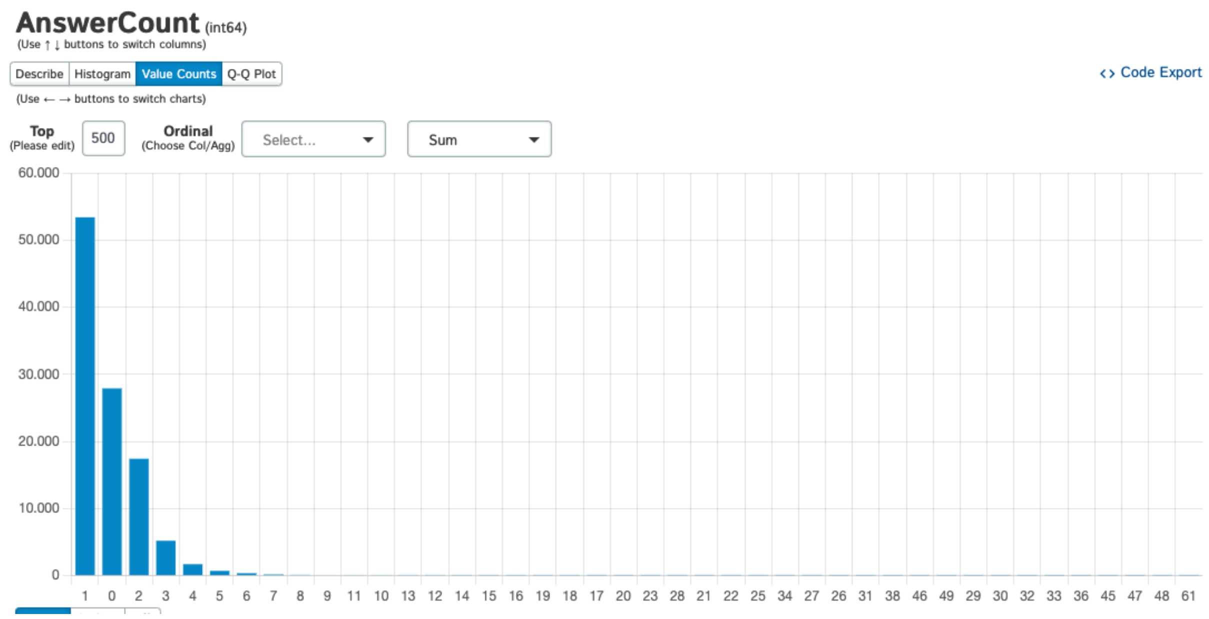


Figure 15 – Value of Answer Count

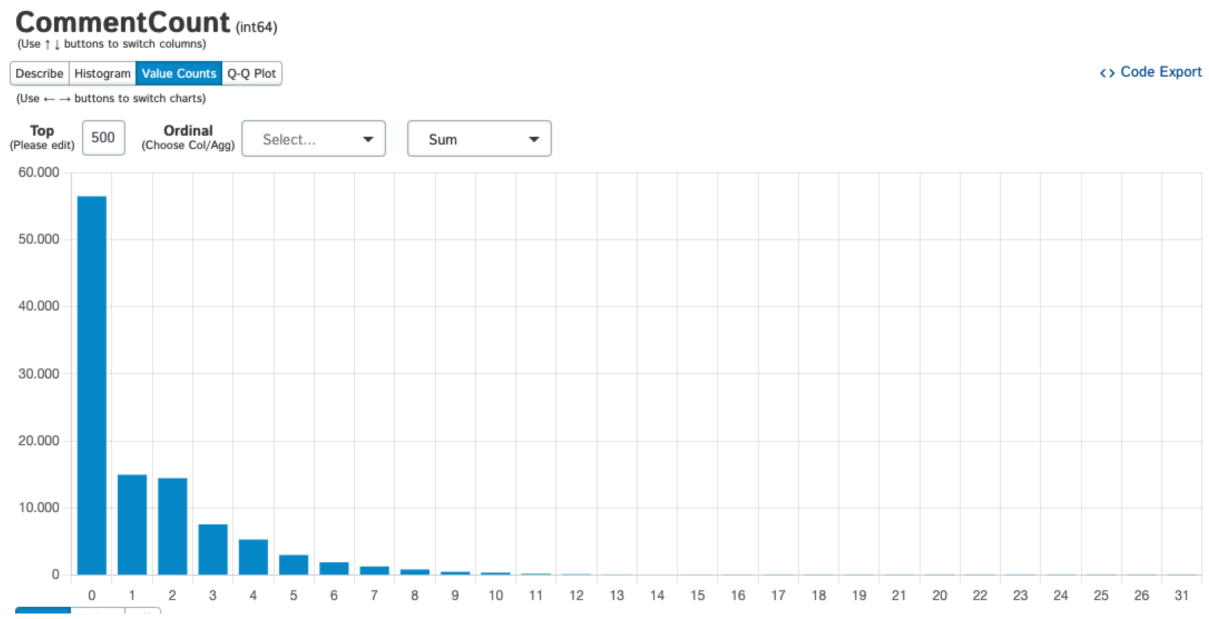


Figure 16 – Value of Comment Count

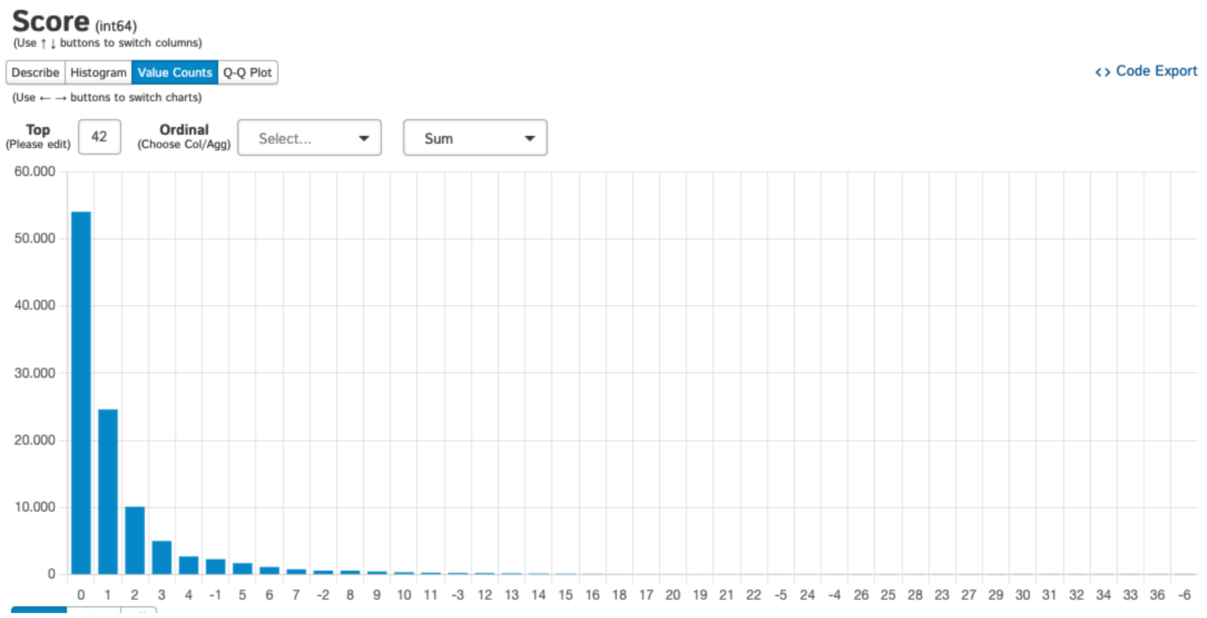


Figure 17 – Value of Score

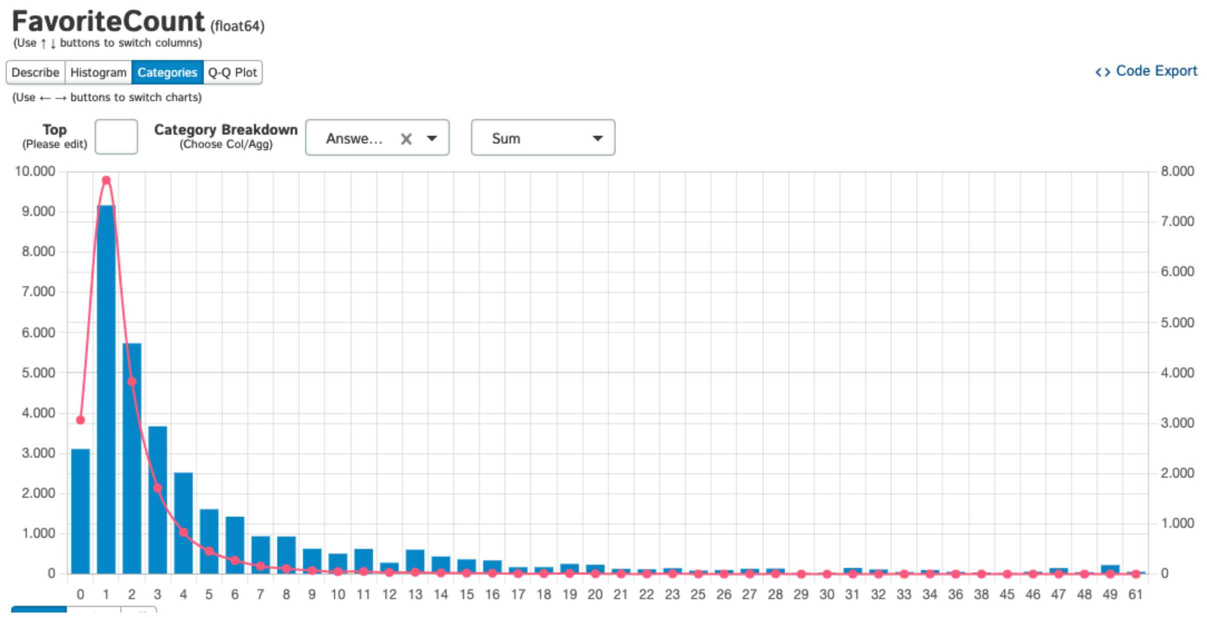


Figure 18 – Value of Favorite Count

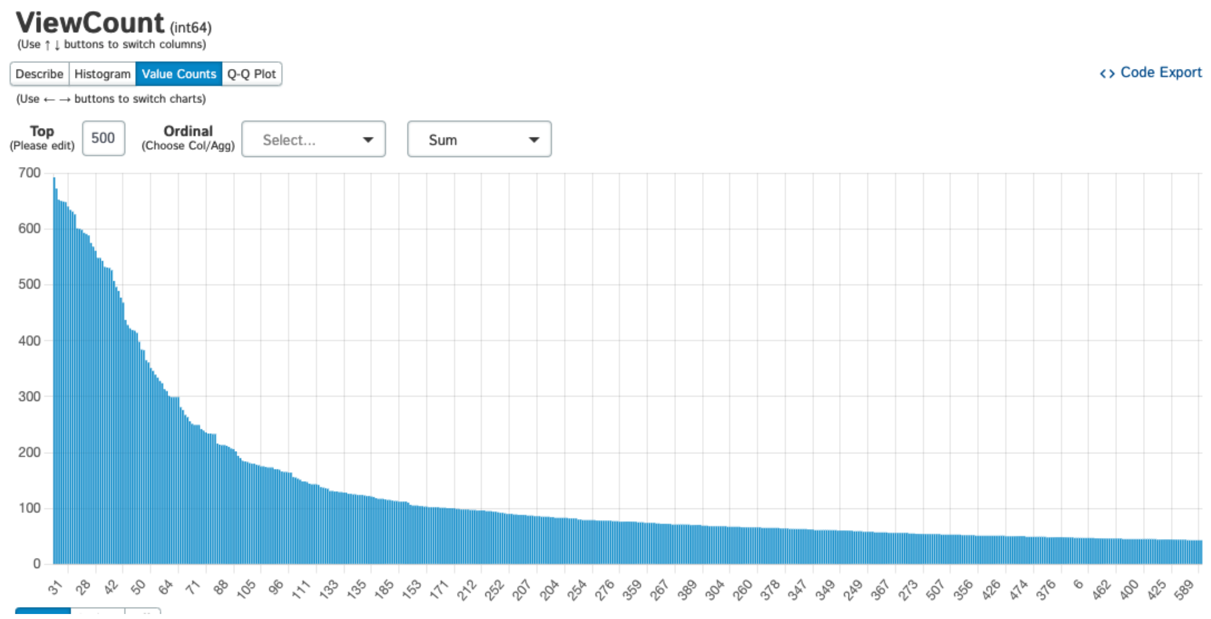


Figure 19 – Value of View Count

4.1.4 Preliminary Discussion

Analyzing the generated graphs on this preliminary analysis is possible to see different topics on the word clouds. It is possible to see discussions on different aspects of React Native development. An example is the navigation topic that can be seen on all word clouds. It is possible to assume that it is difficult for the developers to navigate between the created pages. Another example that appears on Titles' and the Tags' word clouds is Redux, a tool to help the React Native developer save states on the application to be accessed and updated by the whole app. Its appearance on the word clouds means that many developers have doubts about its use.

The fact that it is possible to see different aspects of React Native development on the word clouds generated means that specific words are used on the questions that categorize the discussion's topics. The Mallet tool will use those words to separate the questions into those topics seen in the word clouds.

On the value analysis, it is possible to see which values are used on each attribute. Most of them use a number between 0 and 10. However, some attributes take more significant values. An example is the view count that reaches more than a hundred visualizations.

4.2 Understand the Scoccia, Migliarini e Autili (2021) project repository

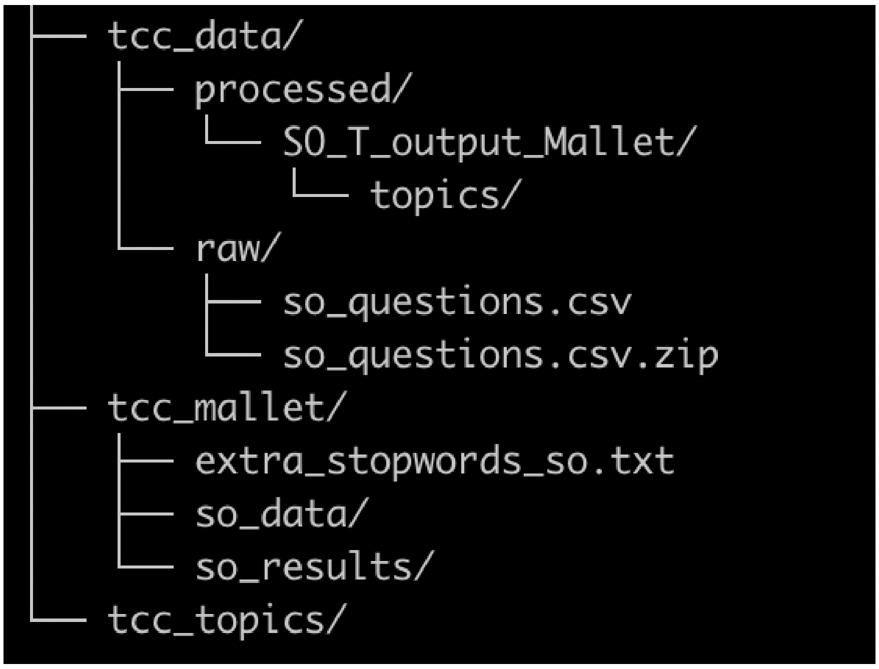
The Scoccia, Migliarini e Autili (2021) project repository was forked to run experiments and make the contributions for this study. The forked repository is at the link https://github.com/gabibguedes/MSR2021Replication/.

4.2.1 Apply mallet on the React Native dataset

This experiment was made to understand the project structure and get a first look at how to adapt it to receive different inputs. This first experiment aims to get the first division of topics of the react-native dataset.

4.2.1.1 Create similar file tree to run experiment

The Scoccia, Migliarini e Autili (2021) repository uses a folder structure to store their dataset and save their output files to run the experiments. A new file tree was created to start a new experiment with this project. This new file tree is similar to the original one to store the dataset and the generated outputs.



```
├── tcc_data/
│   ├── processed/
│   │   └── SO_T_output_Mallet/
│   │       └── topics/
│   └── raw/
│       ├── so_questions.csv
│       └── so_questions.csv.zip
├── tcc_mallet/
│   ├── extra_stopwords_so.txt
│   ├── so_data/
│   └── so_results/
└── tcc_topics/
```

Figure 20 – Added file tree

The project's dataset was added at **tcc_data/raw/so_questions.csv**. All scripts used in the **notebooks/** folder were adapted to fit this experiment's file tree.

4.2.1.2 Clean the datasets data

The first step was to clean the dataset's data using the **clean_stackoverflow_data.py** script. Entering the **notebooks/** folder and running the following command:

```
python3 clean_stackoverflow_data.py
```

Listing 4 – Runs script to clean the dataset

This script generated another dataset CSV file without HTML tags and stopwords at **tcc_data/processed/so_questions.csv**.

4.2.1.3 Export dataset to Mallet

The Mallet tool needs the questions input to be separated into different documents. The **export_so_to_mallet.py** script creates **.txt** files for all questions on the CSV dataset.

For this experiment, the files were created using the questions title and body. The script was run on the command line, inside the **notebooks/** folder.

```
python3 export_so_to_mallet.py
```

Listing 5 – Runs script to export dataset to Mallet

The files for each questions were stored at the **tcc_mallet/so_data/** folder. The file names are the question's IDs.

4.2.1.4 Run Mallet

The following mallet commands were used with the questions separated into different files.

```
mallet/mallet-2.0.8/bin/mallet import-dir --input tcc_mallet/so_data/ --
    output tcc_mallet/so.mallet --keep-sequence --remove-stopwords --
    extra-stopwords tcc_mallet/extra_stopwords_so.txt

mallet/mallet-2.0.8/bin/mallet train-topics --random-seed 100 --input
    tcc_mallet/so.mallet --num-topics 15 --optimize-interval 20 --output-
    state tcc_mallet/so-topic-state.gz --output-topic-keys tcc_mallet/
    so_keys.txt --output-doc-topics tcc_mallet/so_composition.txt --
    diagnostics-file tcc_mallet/so_results/so_diagnostics.xml
```

Listing 6 – Mallet commands

4.2.1.5 Understand the mallet output

To understand the mallet output the following script was used to create **.csv** tables to separate the files in the topics using the output generated by the mallet tool.

```
python3 parse_topics_composition.py
```

Listing 7 – Runs script to parse Mallet's result

A script was created to place all the topic's content into one file to compare and understand the similarities and evaluate the generated topics.

```
import pandas as pd

def df_to_file(row, topic):
    file_name = str(int(row['filename']))

    text_file = open("../tcc_mallet/so_data/{}.txt".format(file_name), "r")
    content = text_file.read()
    text_file.close()

    path = '../tcc_topics/' + topic + '.txt'

    with open(path, 'a+') as file:
      file.write(content + '\n\n')

print('Uniting documents...')

for i in range(1,15):
  topic = 'topic_{}'.format(i)
  print(topic)
  so = pd.read_csv('../tcc_data/processed/SO_T_output_Mallet/topics/{}.csv'.format(topic))
  for index, row in so.iterrows():
    df_to_file(row,  topic)

print('Done!')
```

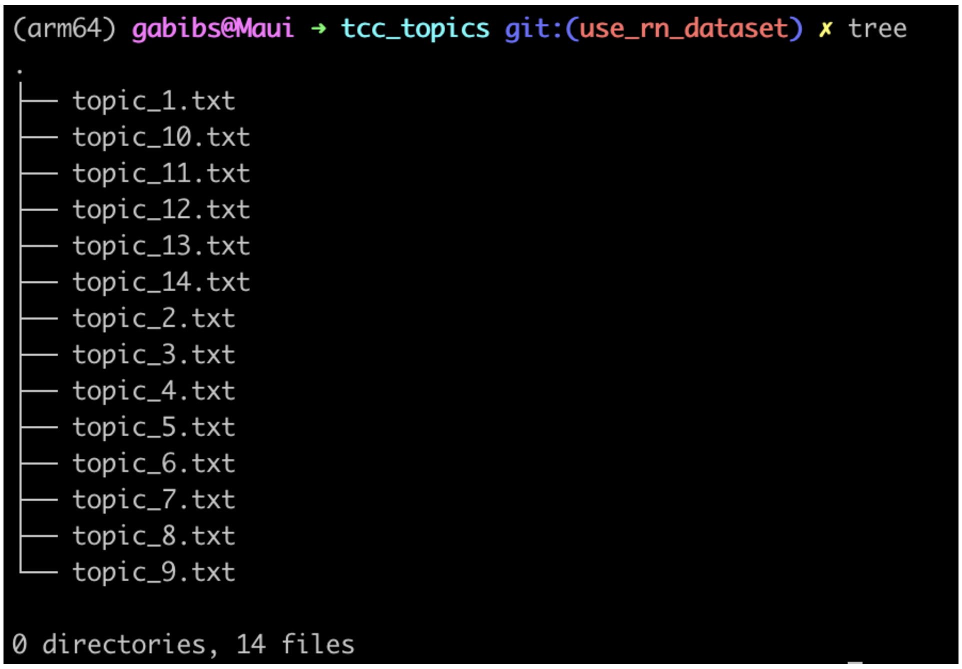
Listing 8 – Created script to gather topic's documents

The following command was used to run the created script.

```
python3 unite_topics_in_one_file.py
```

Listing 9 – Runs the script to unite topic's question in one file

This script's result is demonstrated in the picture below (21). All questions under one topic are placed into one file, creating N files for N topics generated.



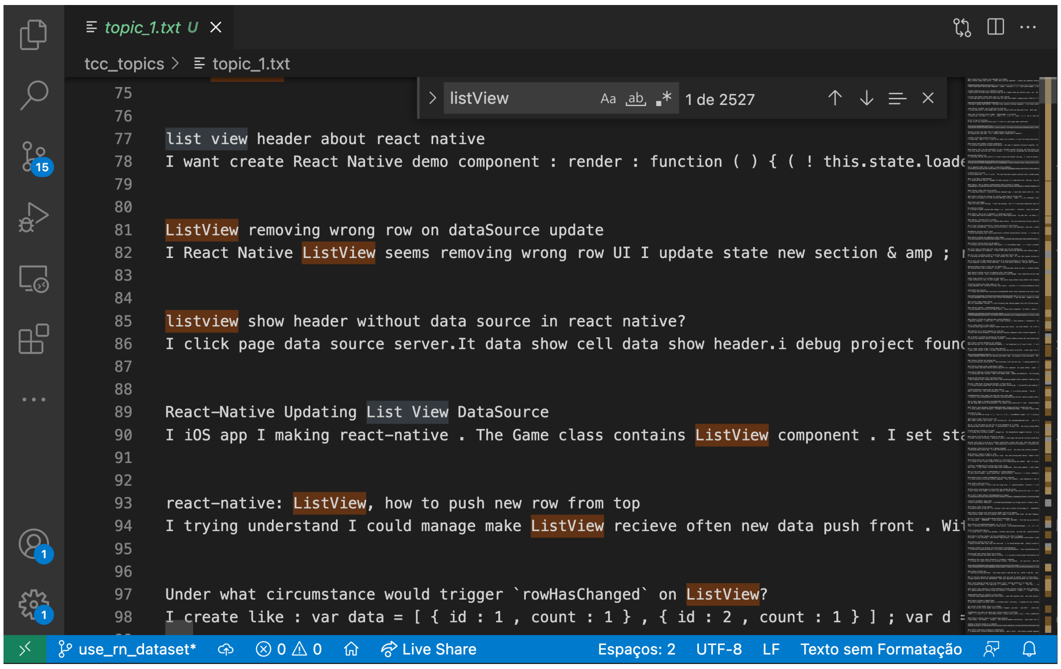
```
(arm64) gabibs@Maui ➜ tcc_topics git:(use_rn_dataset) ✗ tree
.
├── topic_1.txt
├── topic_10.txt
├── topic_11.txt
├── topic_12.txt
├── topic_13.txt
├── topic_14.txt
├── topic_2.txt
├── topic_3.txt
├── topic_4.txt
├── topic_5.txt
├── topic_6.txt
├── topic_7.txt
├── topic_8.txt
└── topic_9.txt

0 directories, 14 files
```

Figure 21 – Topic files generated

4.2.1.6 Results

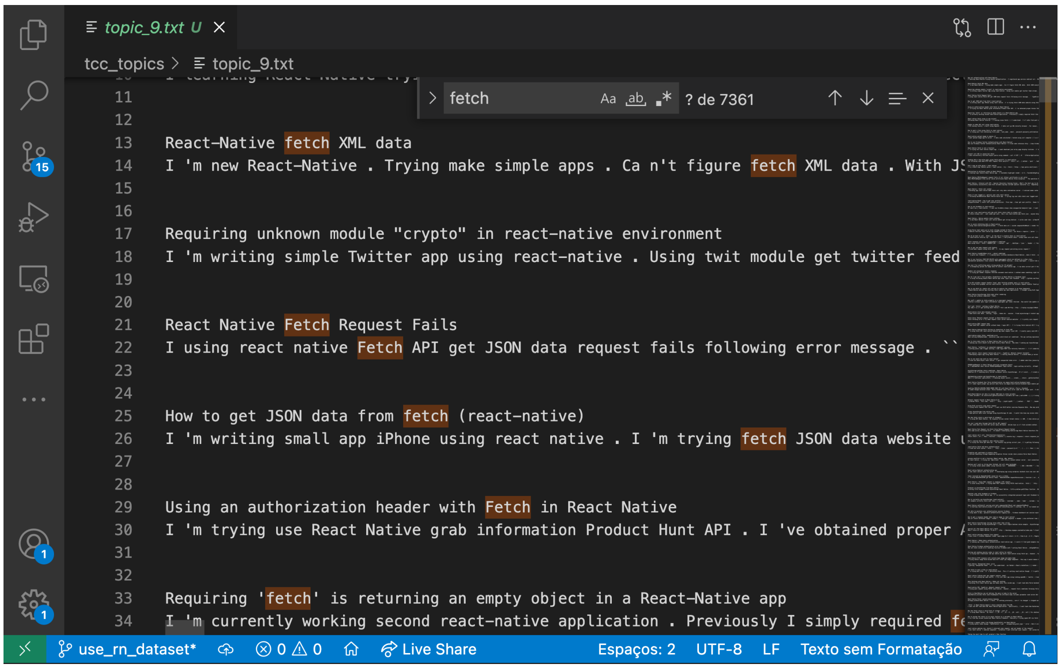
Running this experiment made the project more understandable. Reading the script and comprehending its purpose and how this project operates was necessary. After executing the project, it was possible to see different contexts in each topic file while analyzing the result of the topic division. The pictures below (22, 23, 24, 25) show some of the topics identified.

It is possible to identify some contexts in some document files. As in Figure 22, the topic determined by the algorithm is ListViews, a react component. All questions seen in this topic file are related to this subject. In 23, all questions are related to the Fetch function, a JavaScript function to make API requests. In 24 the questions are about the app's navigation. And in 25 the questions are related to images and media.



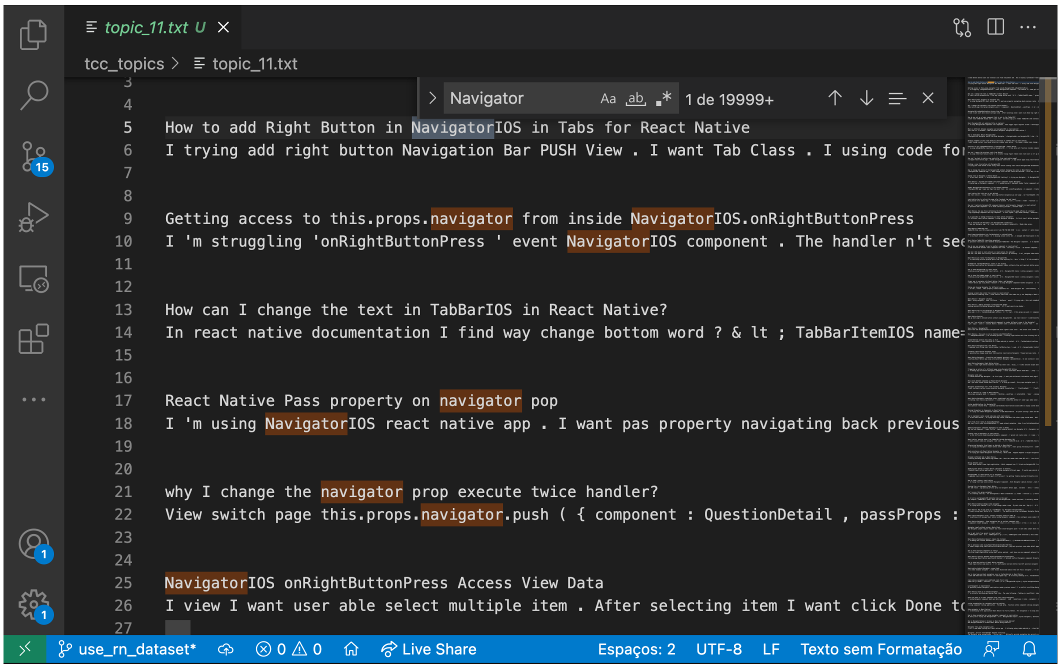
```
list view header about react native
I want create React Native demo component : render : function ( ) { ( ! this.state.loade


ListView removing wrong row on dataSource update
I React Native ListView seems removing wrong row UI I update state new section & amp ;


listview show header without data source in react native?
I click page data source server.It data show cell data show header.i debug project found


React-Native Updating List View DataSource
I iOS app I making react-native . The Game class contains ListView component . I set sta


react-native: ListView, how to push new row from top
I trying understand I could manage make ListView recieve often new data push front . Wit


Under what circumstance would trigger `rowHasChanged` on ListView?
I create like : var data = [ { id : 1 , count : 1 } , { id : 2 , count : 1 } ] ; var d =
```

Figure 22 – Topic 1



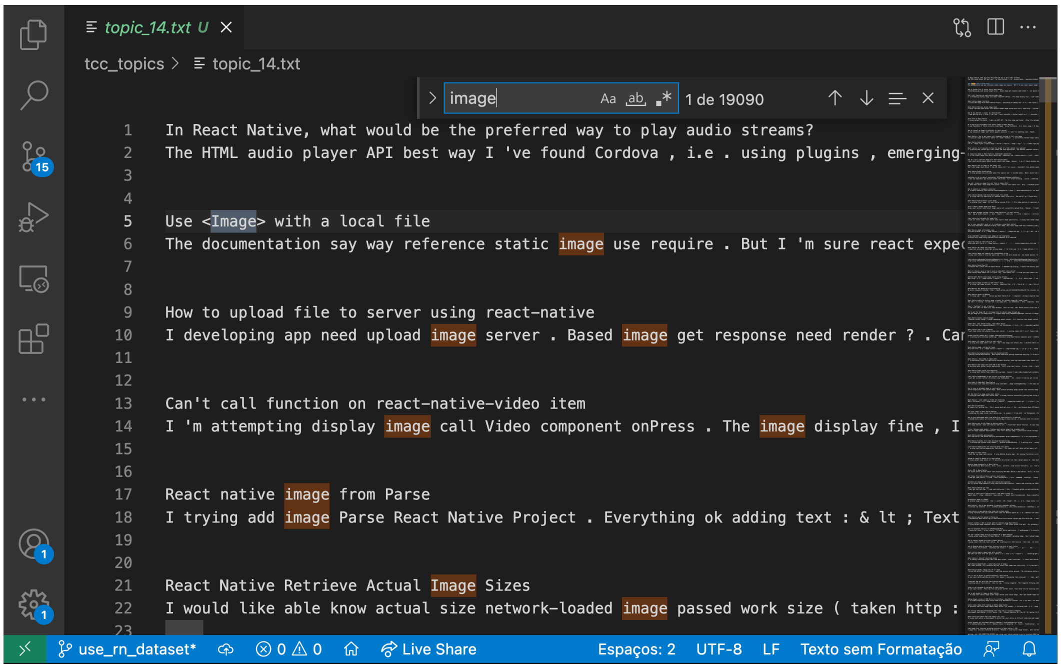
```

React-Native fetch XML data
I 'm new React-Native . Trying make simple apps . Ca n't figure fetch XML data . With JS


Requiring unknown module "crypto" in react-native environment
I 'm writing simple Twitter app using react-native . Using twit module get twitter feed


React Native Fetch Request Fails
I using react native Fetch API get JSON data request fails following error message . ``


How to get JSON data from fetch (react-native)
I 'm writing small app iPhone using react native . I 'm trying fetch JSON data website u


Using an authorization header with Fetch in React Native
I 'm trying use React Native grab information Product Hunt API . I 've obtained proper A


Requiring 'fetch' is returning an empty object in a React-Native app
I 'm currently working second react-native application . Previously I simply required fe
```

Figure 23 – Topic 9



```
How to add Right Button in NavigatorIOS in Tabs for React Native
I trying add right button Navigation Bar PUSH View . I want Tab Class . I using code fo


Getting access to this.props.navigator from inside NavigatorIOS.onRightButtonPress
I 'm struggling 'onRightButtonPress ' event NavigatorIOS component . The handler n't se


How can I change the text in TabBarIOS in React Native?
In react native documentation I find way change bottom word ? & lt ; TabBarItemIOS name=


React Native Pass property on navigator pop
I 'm using NavigatorIOS react native app . I want pas property navigating back previous


why I change the navigator prop execute twice handler?
View switch page this.props.navigator.push ( { component : QuestionDetail , passProps :


NavigatorIOS onRightButtonPress Access View Data
I view I want user able select multiple item . After selecting item I want click Done t
```

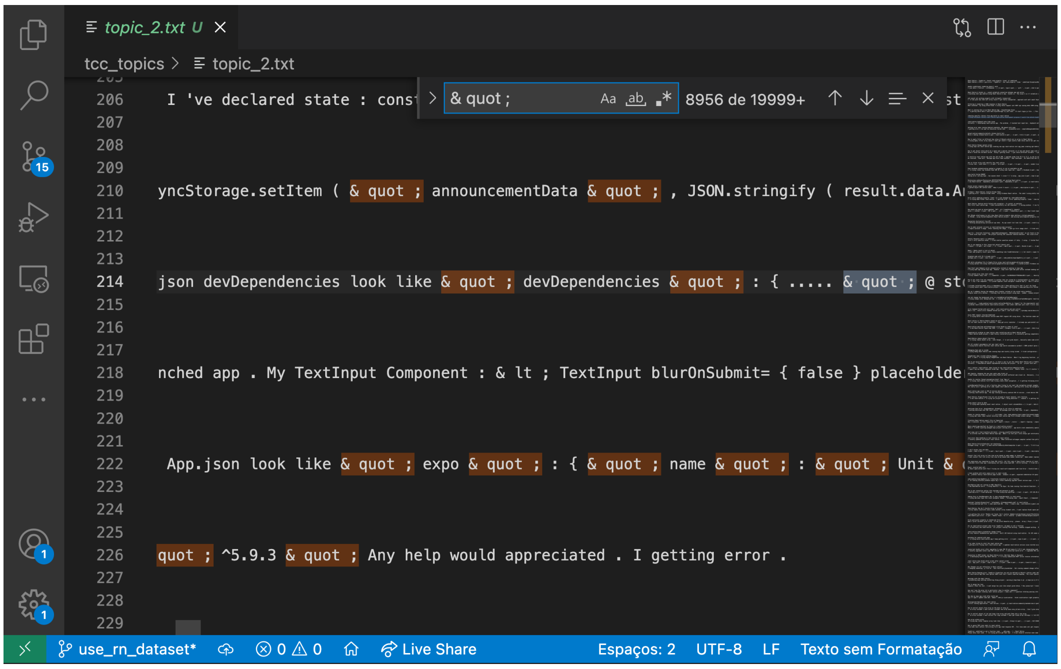
Figure 24 – Topic 11



```
In React Native, what would be the preferred way to play audio streams?
The HTML audio player API best way I 've found Cordova , i.e . using plugins , emerging-


Use <Image> with a local file
The documentation say way reference static image use require . But I 'm sure react expe


How to upload file to server using react-native
I developing app need upload image server . Based image get response need render ? . Ca


Can't call function on react-native-video item
I 'm attempting display image call Video component onPress . The image display fine , I


React native image from Parse
I trying add image Parse React Native Project . Everything ok adding text : & lt ; Text


React Native Retrieve Actual Image Sizes
I would like able know actual size network-loaded image passed work size ( taken http :
```

Figure 25 – Topic 14

Some topics aren't clear enough, meaning that the number of topics configured needs to be calibrated, and there are still some words to be added to the stop words list, as they are influencing the results and don't mean much to this study. As can be seen, the picture below (26) shows a common quote tag identifies a topic.



```
 I 've declared state : cons



yncStorage.setItem ( & quot ; announcementData & quot ; , JSON.stringify ( result.data.A



json devDependencies look like & quot ; devDependencies & quot ; : { ..... & quot ; @ st



nched app . My TextInput Component : & lt ; TextInput blurOnSubmit= { false } placeholde



 App.json look like & quot ; expo & quot ; : { & quot ; name & quot ; : & quot ; Unit &



quot ; ^5.9.3 & quot ; Any help would appreciated . I getting error .
```

Figure 26 – Topic 2

4.3 Evolve the Scoccia, Migliarini e Autili (2021) project repository

The same fork used in the previous section was used to evolve the Scoccia, Migliarini e Autili (2021) repository project (https://github.com/gabibguedes/MSR2021Replication/). A Pull Request was made for this contribution. The Pull Request can be accessed at https://github.com/gianlucascoccia/MSR2021Replication/pull/2.

4.3.1 Custumize inputs and outputs

4.3.1.1 Refactor code to accept different inputs

The next step towards evolving the Scoccia, Migliarini e Autili (2021) code was to adapt it to accept different dataset inputs. The original code contains a hardcoded reference to the path where the dataset is located. The reference was removed to use only an environment variable that the user can set manually. The command below saved the dataset path as an environment variable.

```
export DATASET_PATH=./tcc/so_questions.csv
```

Listing 10 – Exports the dataset path

The **clean_stackoverflow_data.py** script was changed, using the environment variable instead of the hardcoded path. The script followed the example below to access the environment variable.

```
import pandas as pd
import os
DATASET_PATH = os.getenv('DATASET_PATH')

pd.read_csv(DATASET_PATH) # Example of how the dataset is accessed
```

Listing 11 – Access environment variable for dataset path

The number of topics was also hardcoded in some scripts, which were replaced to use en environment variable. The Listing bellow shows how this variable was exported using the command line.

```
export TOPICS_NUM=15
```

Listing 12 – Exports the number of topics

And the scripts were adapted to use this variable following the example.

```
import os

TOPICS_NUM = int(os.getenv('TOPICS_NUM'))
```

Listing 13 – Access environment variable for number of topics

Besides path to the dataset and the number of topics, the scripts also contained hardcoded paths to outputs files, and those were also changed to environment variables making the project more customizable. The command below was used to export the output file as an environment variable.

```
export OUTPUT_PATH=output/
```

Listing 14 – Exports the output path

The scripts were adapted to use the folder from the environment variable and call a new function (Listing 15) to create the subfolders where each output will be placed. The scripts were adapted to call this function to get the subfolder path and ensure it exists.

```
import os

OUTPUT_PATH = os.getenv('OUTPUT_PATH')

def get_output_folder(folder_name):
  subfolder = os.path.join(OUTPUT_PATH, folder_name)

  if not os.path.exists(subfolder):
    os.makedirs(subfolder)
    print('Folder {} created!'.format(subfolder))

  return subfolder
```

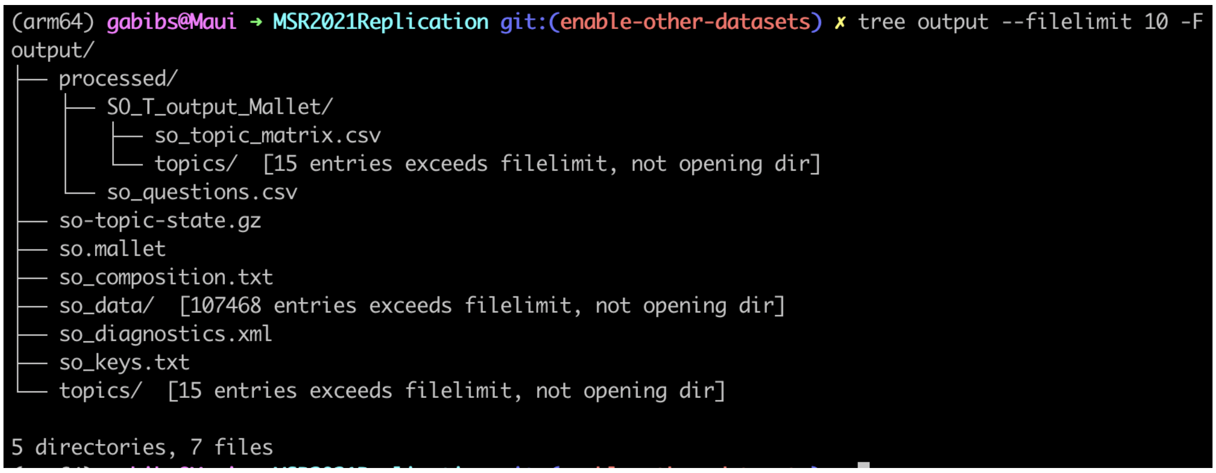
Listing 15 – Access environment variable for output path and create subfolder

On the Listing bellow there is an example of how this function is called in the scripts.

```python
from output_folder import get_output_file , get_output_folder

OUT_FOLDER = get_output_folder('so_data/')
```

Listing 16 – Call to get_output_folder function to ensure the output path

After running the project scripts, the output folder contains all outputs generated, as shown in Figure 27.



```
(arm64) gabibs@Maui ➜ MSR2021Replication git:(enable-other-datasets) ✗ tree output --filelimit 10 -F
output/
├── processed/
│   ├── SO_T_output_Mallet/
│   │   ├── so_topic_matrix.csv
│   │   └── topics/  [15 entries exceeds filelimit, not opening dir]
│   └── so_questions.csv
├── so-topic-state.gz
├── so.mallet
├── so_composition.txt
├── so_data/  [107468 entries exceeds filelimit, not opening dir]
├── so_diagnostics.xml
├── so_keys.txt
└── topics/  [15 entries exceeds filelimit, not opening dir]

5 directories, 7 files
```

Figure 27 – Output folder file tree

Considering the original file tree is to no longer be used, the files for the extra stopwords, both for GitHub and Stack Overflow dataset, were moved to another folder called **extra_stopwords/** on the repository root.

4.3.2 Refactor the scripts into modules

The Scoccia, Migliarini e Autili (2021) project scripts are not componentized, as it was implemented for the whole file to be executed. This next step for evolving the project was to refactor the scripts into modules so that other scripts could import them, making the code behind it more abstract.

In this part of the contribution, the scripts were refactored to be componentized into functions. The main function was also added, maintaining the script execution.

Two new scripts were created. One script is to call all the steps necessary to clean the Stack Overflow dataset, named **prepare_dataset.py** (Listing 17) and the other, named **manage_results.py**, to interpret the results (Listing 18).

```python
from clean_stackoverflow_data import clean_so_data
from export_so_to_mallet import export_to_mallet

if __name__ == '__main__':
    print('Cleaning dataset...')
    clean_so_data()
    print('Exporting documents to Mallet...')
    export_to_mallet()
    print('Done!')
```

Listing 17 – Script to clean Stack Overflow dataset

```python
from parse_topics_composition import parse_topics
from unite_topics_in_one_file import unite_questions_documents_by_topic

if __name__ == '__main__':
  print('Parsing topics...')
  parse_topics()
  print('\nUniting questions by topic...')
  unite_questions_documents_by_topic()
```

Listing 18 – Script to interpret the Mallet results

With those scripts, the new steps to execute the project are:

1. Export the dataset input, the output path and the number of topics as environment variables

2. Run the script to clean the stack overflow dataset

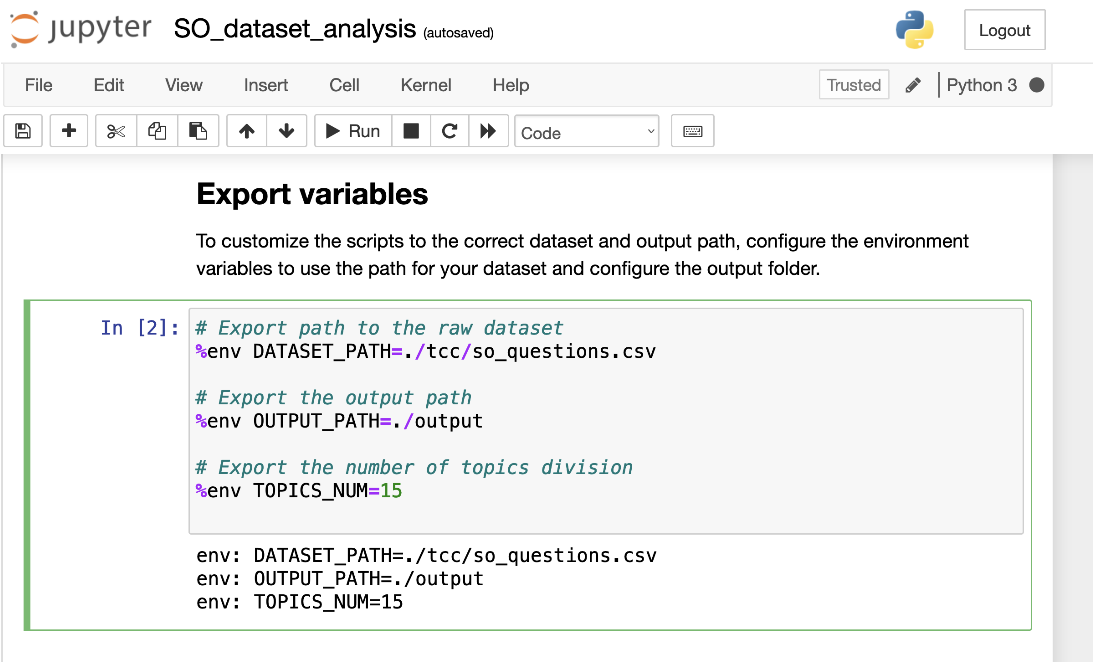
3. Run the Mallet commands

4. Run the script to interpret the results

4.3.3 Create a notebook to run the steps

This project is complicated, and part of this study is to make it easier for other developers to use and understand. The last contribution step was creating a Jupyter Notebook with all steps and explanations to execute the project. Notebooks are a great tool because they contain cells that support markdown to the explanations and have cells that can execute python and bash codes to run the scripts, which will be necessary to run the code with different inputs.

The notebook was made based on the created scripts (Listing 17 and 18), adding instructions and explanations to what happens in each step. The user is instructed to export the correct path for the dataset used and the notebook's output folder at the beginning. After configuring the environment variables, the notebook follows the scripts to clean the dataset, run the mallet and run the scripts to parse the results. See the pictures below (28, 29, 30 and 31).


jupyter SO_dataset_analysis (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code

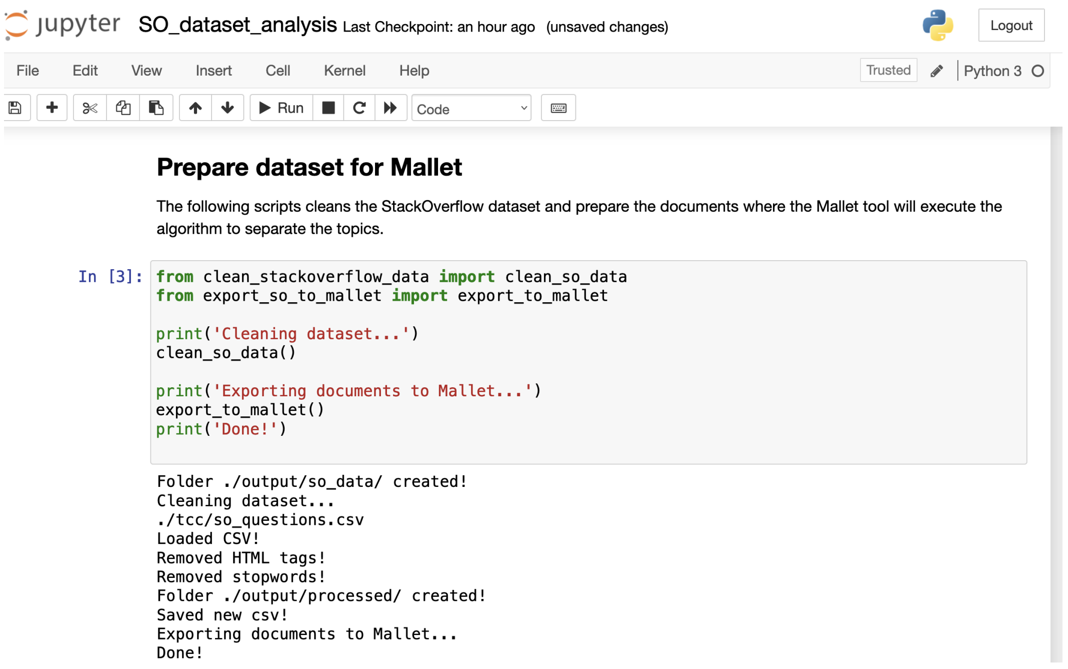
Export variables

To customize the scripts to the correct dataset and output path, configure the environment variables to use the path for your dataset and configure the output folder.

```
In [2]: # Export path to the raw dataset
        %env DATASET_PATH=./tcc/so_questions.csv

        # Export the output path
        %env OUTPUT_PATH=./output

        # Export the number of topics division
        %env TOPICS_NUM=15
```

```
env: DATASET_PATH=./tcc/so_questions.csv
env: OUTPUT_PATH=./output
env: TOPICS_NUM=15
```

Figure 28 – Notebook exporting variables cell


jupyter SO_dataset_analysis Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code

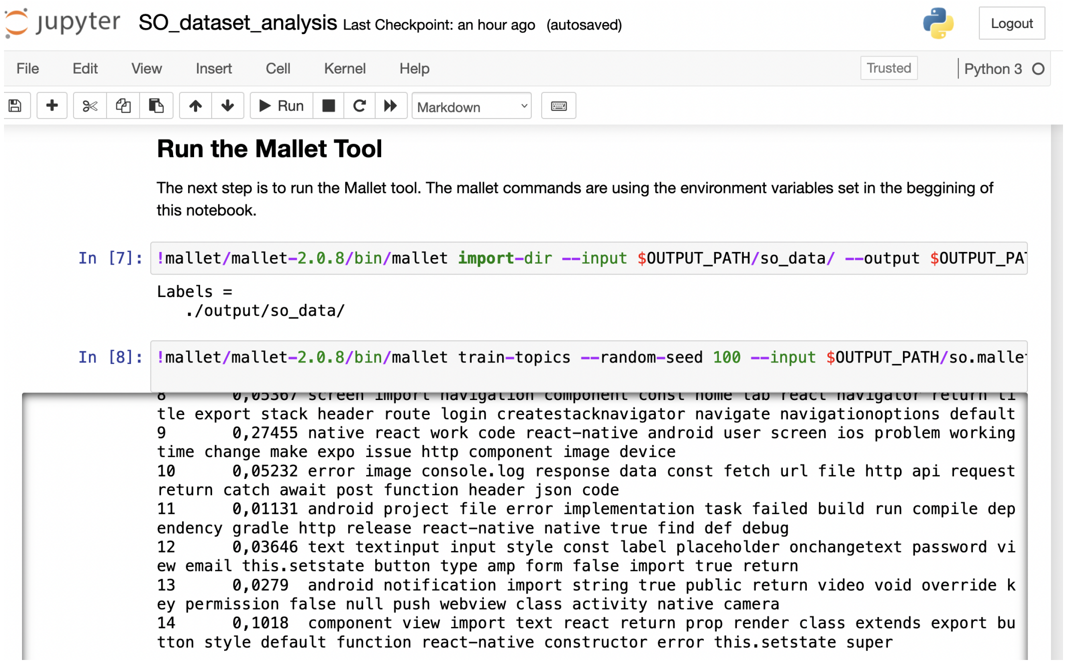
Prepare dataset for Mallet

The following scripts cleans the StackOverflow dataset and prepare the documents where the Mallet tool will execute the algorithm to separate the topics.

```
In [3]: from clean_stackoverflow_data import clean_so_data
        from export_so_to_mallet import export_to_mallet

        print('Cleaning dataset...')
        clean_so_data()

        print('Exporting documents to Mallet...')
        export_to_mallet()
        print('Done!')
```

```
Folder ./output/so_data/ created!
Cleaning dataset...
./tcc/so_questions.csv
Loaded CSV!
Removed HTML tags!
Removed stopwords!
Folder ./output/processed/ created!
Saved new csv!
Exporting documents to Mallet...
Done!
```

Figure 29 – Notebook prepare dataset cell

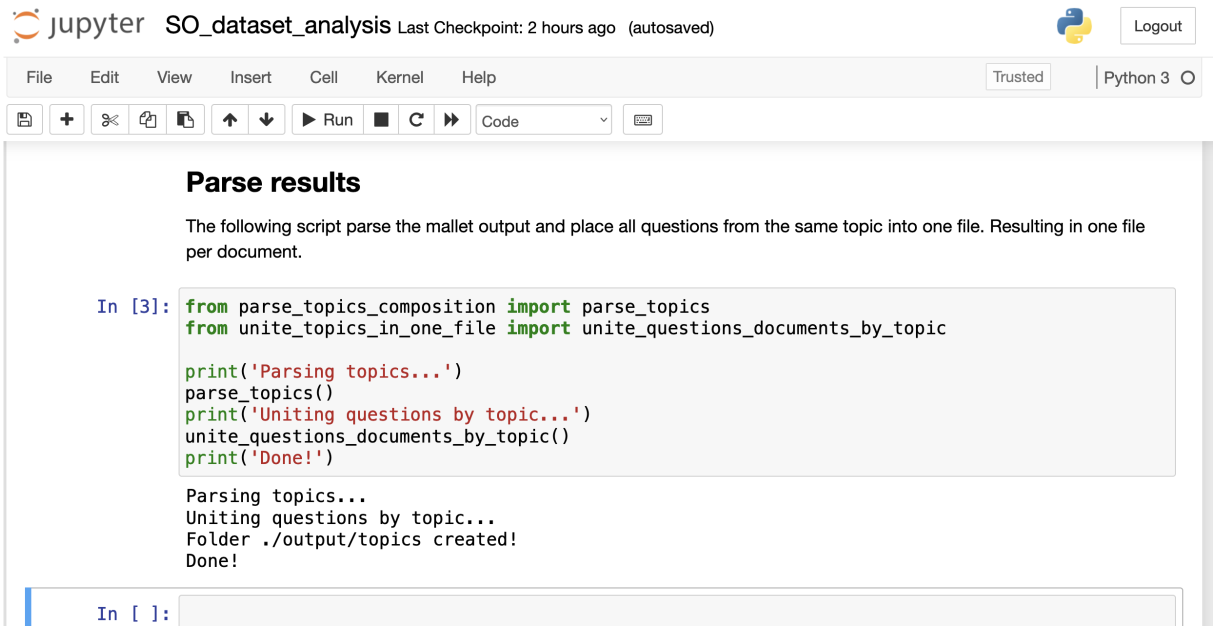

jupyter SO_dataset_analysis Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Markdown

Run the Mallet Tool

The next step is to run the Mallet tool. The mallet commands are using the environment variables set in the beggining of this notebook.

```
In [7]: !mallet/mallet-2.0.8/bin/mallet import-dir --input $OUTPUT_PATH/so_data/ --output $OUTPUT_PA

        Labels =
           ./output/so_data/

In [8]: !mallet/mallet-2.0.8/bin/mallet train-topics --random-seed 100 --input $OUTPUT_PATH/so.malle

        8       0,05367 screen import navigation component const home tab react navigator return ti
        tle export stack header route login createstacknavigator navigate navigationoptions default
        9       0,27455 native react work code react-native android user screen ios problem working
        time change make expo issue http component image device
        10      0,05232 error image console.log response data const fetch url file http api request
        return catch await post function header json code
        11      0,01131 android project file error implementation task failed build run compile dep
        endency gradle http release react-native native true find def debug
        12      0,03646 text textinput input style const label placeholder onchangetext password vi
        ew email this.setstate button type amp form false import true return
        13      0,0279  android notification import string true public return video void override k
        ey permission false null push webview class activity native camera
        14      0,1018  component view import text react return prop render class extends export bu
        tton style default function react-native constructor error this.setstate super
```

Figure 30 – Notebook run mallet commands cell


jupyter SO_dataset_analysis Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code

Parse results

The following script parse the mallet output and place all questions from the same topic into one file. Resulting in one file per document.

```
In [3]: from parse_topics_composition import parse_topics
        from unite_topics_in_one_file import unite_questions_documents_by_topic

        print('Parsing topics...')
        parse_topics()
        print('Uniting questions by topic...')
        unite_questions_documents_by_topic()
        print('Done!')

        Parsing topics...
        Uniting questions by topic...
        Folder ./output/topics created!
        Done!

In [ ]:
```

Figure 31 – Notebook parse results cell

Considering the original project used both Stack Overflow and GitHub datasets, some scrips are prepared to deal with a Stack Overflow dataset and others with GitHub. One dataset cannot be used in all scripts. This notebook can only be used with a Stack Overflow dataset.

4.4 Calibrating and running the React Native dataset

After being implemented, the notebook was used to rerun the algorithm for the react-native dataset. As concluded in the 4.2.1, the algorithm needed to calibrate before running the react-native dataset and get better results. The first run was considered to calibrate the algorithm, as adding some stopwords that weren't being counted before was necessary.

Stopwords were added, and the notebook was used to rerun the scripts. The sections below describes the experiments made with different number of topics, analyzing the context of each topic created. The subjects were analyzed by the first five questions of each topic's document.

4.4.1 10 topics

Table 2 represents the analysis made in the results with 10 topics.

Topic	Context
Topic 1	Problems related to View scroll.
Topic 2	AwesomeProject setup problems, an example project
Topic 3	Unclear
Topic 4	App's navigation
Topic 5	Application styling
Topic 6	Android app not building
Topic 7	Instalation, first run questions and some media handling problems
Topic 8	TextInput component
Topic 9	Unclear
Topic 10	Fecth and authentication to connect to other services APIs

Table 2 – 10 Topics division context

4.4.2 15 topics

Table 3 represents the analysis made in the results with 15 topics.

Topic	Context
Topic 1	Maps and geolocation
Topic 2	Animation
Topic 3	Redux and DateString
Topic 4	Image and View Dimensions
Topic 5	AwesomeProject, used to setup the development environment
Topic 6	ListView component
Topic 7	Instalation problems
Topic 8	Authentication in outside services
Topic 9	App's navigation
Topic 10	Instalation, first run questions and some media handling problems
Topic 11	Connecting to APIs and using the Fetch function
Topic 12	Android app not building
Topic 13	TextInput component
Topic 14	Unclear
Topic 15	Unclear

Table 3 – 15 Topics division context

4.4.3 20 topics

Table 4 represents the analysis made in the results with 20 topics.

Topic	Context
Topic 1	Unclear
Topic 2	Initial installation
Topic 3	Unclear
Topic 4	Android app not building
Topic 5	TextInput component
Topic 6	API calls and outside services authentication. This topic is a little unclear.
Topic 7	Unclear
Topic 8	WebView component
Topic 9	Redux
Topic 10	Integration with Camera, and other similar native problems. More iOS related.
Topic 11	Map view, coordinates and geolocation
Topic 12	Unclear
Topic 13	Apps navigation
Topic 14	Unclear
Topic 15	View styles and other UI problems
Topic 16	Instalation and first run questions
Topic 17	Date string and date picker
Topic 18	Other style problems
Topic 19	ListView component
Topic 20	More app's navigation problems

Table 4 – 20 Topics division context

4.4.4 Result

After uniting all the clear topics division from the experiments with 10, 15, and 20 topics, removing the unclear divisions, and separating the mixed contexts identified, we have Table 5 as a result of this analysis for the React Native discussed topics in StackOverflow.

Topics
AwesomeProject setup problems, an example project
Installation
First run questions
Media handling problems
Problems related to View scroll.
ListView component
App's navigation
Application styling
Android app not building
TextInput component
Maps and geolocation
Animation
Redux
DateString and DatePicker
Image and View Dimensions
Authentication in outside services
Connecting to APIs using the Fetch function
WebView component
Integration with Camera, and other similar native problems. More iOS related.

Table 5 – Topics identified

5 Conclusion

The contribution to the original project made it easier to run the script with other datasets, and the possibility to differentiate the output result made it easier to analyze the run on a different number of topics. Also, running the experiments was an excellent opportunity to understand the project better.

The Pull Request was accepted to the Scoccia, Migliarini e Autili (2021) repository and is already available to be used to study Stack Overflow datasets in other contexts. This contribution facilitates understanding the code by bringing documentation on how to use it and get the dataset. The notebooks facilitate the execution of scripts, having all configurations needed on the notebook and the python scripts call simplified.

However, the contribution in this paper is made only for Stack Overflow datasets, while the original project also has scripts to handle Github datasets. A next contribution should include the Github dataset handling into the Jupyter Notebooks. There are other scripts to get metrics on the results that weren't explored in this paper. Also, in a future contribution, other aspects of the Stack Overflow dataset could be used to categorize the topics, such as the number of answers or views.

Running the notebook using the react-native dataset proved that the contribution optimized the use of the scripts and facilitated using the Mallet tool. The notebook ran with 10, 15, and 20 topics division to calibrate the algorithm for the react-native dataset. The experiment running the mallet with 15 topics was the one with a more precise division. However, uniting with the 10 and 20 topics run, we have better understated the issues discussed in the React Native framework.

Bibliography

ABDELLATIF, A. et al. Challenges in chatbot development: A study of stack overflow posts. *Proceedings - 2020 IEEE/ACM 17th International Conference on Mining Software Repositories, MSR 2020*, Association for Computing Machinery, Inc, p. 174–185, 6 2020. Cited on page 25.

BLEI, D. M.; NG, A. Y.; EDU, J. B. Latent dirichlet allocation michael i. jordan. *Journal of Machine Learning Research*, v. 3, p. 993–1022, 2003. Cited 2 times on pages 24 and 30.

DIBONA, C.; OCKMAN, S. *Open sources: Voices from the open source revolution.* [S.l.]: " O'Reilly Media, Inc.", 1999. Cited on page 24.

FUGGETTA, A. Open source software - an evaluation. *Journal of Systems and Software*, v. 66, p. 77–90, 4 2003. ISSN 01641212. Cited on page 24.

Git. *Git*. 2021. Disponível em: <https://git-scm.com/>. Cited on page 31.

GitHub. *React Native Topic*. 2021. Disponível em: <https://github.com/topics/react-native>. Cited on page 22.

GNU. *What is Free Software?* 2022. Disponível em: <https://www.gnu.org/philosophy/free-sw.en.html#f1>. Cited on page 24.

GRAHAM, S.; WEINGART, S.; MILLIGAN, I. *Getting started with topic modeling and MALLET*. [S.l.], 2012. Cited on page 25.

JONES, J. et al. Novel approach to cluster patient-generated data into actionable topics: case study of a web-based breast cancer forum. *JMIR medical informatics*, JMIR Publications Inc., Toronto, Canada, v. 6, n. 4, p. e9162, 2018. Cited on page 25.

Jupyter. *Jupyter*. 2021. Disponível em: <https://jupyter.org/>. Cited 2 times on pages 29 and 32.

MALAVOLTA, I. et al. End users' perception of hybrid mobile apps in the google play store. 2015. Disponível em: <http://cs.gssi.infn.it/ms>. Cited on page 21.

Man Group. *wordcloud*. 2021. Disponível em: <https://pypi.org/project/dtale>. Cited on page 33.

MCCALLUM, A. K. *MALLET: A Machine Learning for Language Toolkit*. 2002. Disponível em: <http://mallet.cs.umass.edu/>. Cited 3 times on pages 24, 30, and 32.

MEIRELLES, P. et al. A students' perspective of native and cross-platform approaches for mobile application development. In: . [S.l.]: Springer Verlag, 2019. v. 11623 LNCS, p. 586–601. ISBN 9783030243074. ISSN 16113349. Cited on page 21.

MUELLER, A. *wordcloud*. 2020. Disponível em: <https://amueller.github.io/word_cloud/index.html>. Cited on page 33.

npm. *React Native*. 2021. Disponível em: <https://www.npmjs.com/package/react-native>. Cited on page 22.

NUNKESSER, R. Beyond web/native/hybrid: A new taxonomy for mobile app development. *Proceedings - International Conference on Software Engineering*, IEEE Computer Society, p. 214–218, 5 2018. ISSN 02705257. Cited 2 times on pages 9 and 20.

Octoverse. *Projects*. 2019. Disponível em: <https://octoverse.github.com/2018/projects#repositories>. Cited on page 22.

Pandas. *Pandas*. 2021. Disponível em: <https://pandas.pydata.org/>. Cited on page 33.

PENZENSTADLER, B. et al. Systematic mapping study on software engineering for sustainability (se4s). In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. [S.l.: s.n.], 2014. p. 1–14. Cited on page 25.

Python. *Python*. 2021. Disponível em: <https://www.python.org/>. Cited 2 times on pages 29 and 32.

React Native. *Core Components and Native Components*. 2021. Disponível em: <https://reactnative.dev/docs/intro-react-native-components>. Cited on page 23.

React Native. *React Native. Learn once, write anywhere*. 2021. Disponível em: <https://reactnative.dev/>. Cited 2 times on pages 17 and 22.

SARICA, S.; LUO, J. Stopwords in technical language processing. *Plos one*, Public Library of Science San Francisco, CA USA, v. 16, n. 8, p. e0254937, 2021. Cited on page 25.

SCOCCIA, G. L.; MIGLIARINI, P.; AUTILI, M. Challenges in developing desktop web apps: a study of stack overflow and github. 2021. Disponível em: <https://gianlucascoccia.github.io/assets/pdf/MSR2021.pdf>. Cited 14 times on pages 5, 7, 16, 17, 25, 27, 30, 43, 45, 47, 49, 51, 53, and 59.

SMUTNý, P. Mobile development tools and cross-platform solutions. In: . [S.l.: s.n.], 2012. p. 653–656. ISBN 9781457718687. Cited on page 20.

Stack Exchange. *Stack Exchange*. 2021. Disponível em: <https://stackexchange.com/>. Cited on page 31.

Stack Exchange Data Explorer. *Stack Exchange Data Explorer*. 2021. Disponível em: <https://data.stackexchange.com/>. Cited on page 28.

Stack Overflow. *2021 Developer Survey*. 2021. Disponível em: <https://insights.stackoverflow.com/survey/2021>. Cited on page 23.

Stack Overflow. *Who we are*. 2021. Disponível em: <https://stackoverflow.com/company>. Cited 2 times on pages 17 and 23.

StatCounter. *Mobile Operating System Market Share Worldwide - Sept 2020 - Aug 2021*. 2021. Disponível em: <https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-202009-202108-bar>. Cited 2 times on pages 9 and 19.

TUN, P. M. Choosing a mobile application development approach. 2014. Disponível em: <https://www.researchgate.net/publication/342437581>. Cited 3 times on pages 9, 21, and 22.

ZOHUD, T.; ZEIN, S. Cross-platform-mobile-app-development-in-industry-a-multiple-case-study. 2021. Cited on page 23.