Final Year Project

Time Series Based Summarization

Mak Yen Wei

Bachelor of Computer Science

(Data Science)

Jan 2023

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Google occupies over ninety percent of people's working hours when it comes to engineering. A large portion of a developer's responsibility is devoted to the removal of errors. Even though it is essential, it might be a tedious waste of time to look through a number of online forums and stack exchanges in order to solve an easy linear equation in Python(Menéndez, 2021).It is not unusual for engineers to explore Stack Overflow for a number of hours in an effort to find an answer that is definite.

It is infamously difficult to address bugs in software due to the vast number of interdependent libraries and packages that are utilised by developers working in a wide variety of programming languages, frameworks, and platforms. Sometimes the problem is caused by a fresh new bug that was included in the most current version of the application. There is also the possibility that a mistake was made in the coding. In rare instances, the problem may also lie inside the local environment of the developer. There might be an infinite number of contributing elements here.

By utilising text processing techniques, ranking approaches, and summarization models, the work of this thesis is to locate the best solution to the problem faced by engineers. With premilinary research, stitching together context from online forums such as stackoverflow with code sharing platforms such as github in order to compile the best possible solution to the problem at hand. This was particularly highlighted in

(Manes & Baysal, 2019).

However, one of the major issue with proposing such framework is the data gathering process. The data is not readily available and hence data scraping is required. To find the best solution, it's benifical to have as much dataset as possible. Throughout the course of this investigation, a framework to address this issue by using cleverly timed cronjobs to scrape data from online forums as well as a FAQ generation process will be presented.

1.1 Problem Statement

FAQ Generation has been a long history, manually summarizing and cherry picking the best anwers to a question was the framework used for decades. The drawback of this method is that it's time consuming and not scalable(). With the advent of deep learning, the problem of FAQ generation has been addressed by using summarization models. **However the research done in the software engineering area is very limited**, most researches have made effort in the banks, insurance FAQ generation ().

Furthermore, **FAQ Generation on technical forums is a challenging task** as the corpus used will differ from general datasets such as the medical field. The terminilogies used differs drastically and hence special care must be taken to ensure the faq generated is accurate. (Sarica & Luo, 2020) (Gerlach, Shi, & Amaral, 2019).

In addition, **keywords used by developers are not always accurate enough to find the best solution**. While google searching does a great job at classifying keywords to finding the matching solution, the same cannot be said for technical forums. Developers are not always able to find the best solution to their problem. (Menéndez, 2021) because the amount of factors that contributes to a problem can be overwhelming.

1.2 Hypothesis

With the advent research gone into summarization, topic modelling, sentiment analysis models. A framework can be developed to address the problem of FAQ generation on technical forums. The framework will be able to generate a FAQ based on the keywords asked by the developer and also be able to rank the best solution to the problem.

1.3 Objective

Through the utilisation of text processing techniques, ranking approaches, and summarization models, providing assistance to developers in their search for answers is worth investigating. An automation towards faq generation in the software engineering domain is the primary aim of this thesis

The project objectives are as follows:

- **To gather** dataset from online forums in the software engineering domain.
- **To propose** a scalable, modular framework for FAQ generation on online forums in the software engineering domain.
- **To evaluate** multiple architectures for FAQ generation.

1.4 Scope

Platform

The stackoverflow forums may be scraped, ranked in terms of their usefulness to the developer's use case, and the top-ranked posts' comments, answers, and responses' comments can be summarised using the framework. As an added bonus, a web app will be built so programmers may use the framework to find the best solution to their problem.

Technologies

- **Web Application**: Python, FastAPI, AWS, ReactJS
- **Scraping**: Selenium
- **Text Processing**: NLTK, Spacy
- **Summarizing**: Tensorflow, Keras

Language

In the context of this thesis, the language used is English. The reason for this is because the majority of the online forums are in English.

CHAPTER 2

LITERATURE REVIEW

The following literature review section highlights the existing work that is related to our research. The work reviewed mostly consists of FAQ Processing and a couple NLP techniques to achieve the goal of our research.

One of the caveats of the existing work is that, most of FAQ processing related work are based on the use on non-engineering fields such as the medical field. Likewise, since our research primarily focuses on the engineering field, we hope to migrate any intuition that we can get from the existing work to our research. Therefore a certain degree of reading has been done in terms of handling technical languages and terminologies.

2.1 FAQ Processing

FAQ retrieval is a crucial task when it comes to question answer pairs rankings (Gupta & Carvalho, 2019). It is a process of retrieving the most relevant questions from a large collection of questions based on a user query. Achieving excellent performance in this task will solve many problems that we've discussed in the previous chapter 1.1. However, most of the traditional methods for FAQ generation are based on the use of extensive manual classification and engineering (Gupta & Carvalho, 2019), this method takes tremendous amount of time and effort to be done. This was particularly highlighted by (Raazaghi, 2015) (Hu, Yu, & Jiau, 2010) (Henß, Monperrus, & Mezini, 2012) (Razzaghi, Minaee, & Ghorbani, 2016). In addition, the performance of the system is also affected by the quality of the manual classification and engineering.

To address the issues brought-up, there are attempts and researches that have done to improve the predecessor. One of the approaches on solving the problem, is to automate the process of FAQ generation using NLP techniques. (Gupta & Carvalho, 2019) has made progress on using deep learning methods such as combining both Deep Matching Networks and Multihop Attention Networks in performing FAQ retrieval. Deep Matching Network also known as (DMN) is a deep learning model that generates a matching scores based of 2 matrices inputs. The matrices are made of the dot product of embeddings of every word of question with every word of answers.While Multihop Attention Networks on the other hand is proven to be effective for reasonings tasks like question answering, which is our focus in this paper (Gupta & Carvalho, 2019). The network uses multiple "hops" of attention to gather information from a given input and

make a prediction. It starts by encoding the input and then uses a decoder network that iteratively attends to different parts of the input, in order to generate an output.

(Raazaghi, 2015) (Jijkoun & de Rijke, 2005) takes an approach where a whole architecture is designed to achieve Auto-Faq-Gen which includes scraping, question construction, a ranking algorithm, and the question generation. The architecture is shown in Figure 2.1.

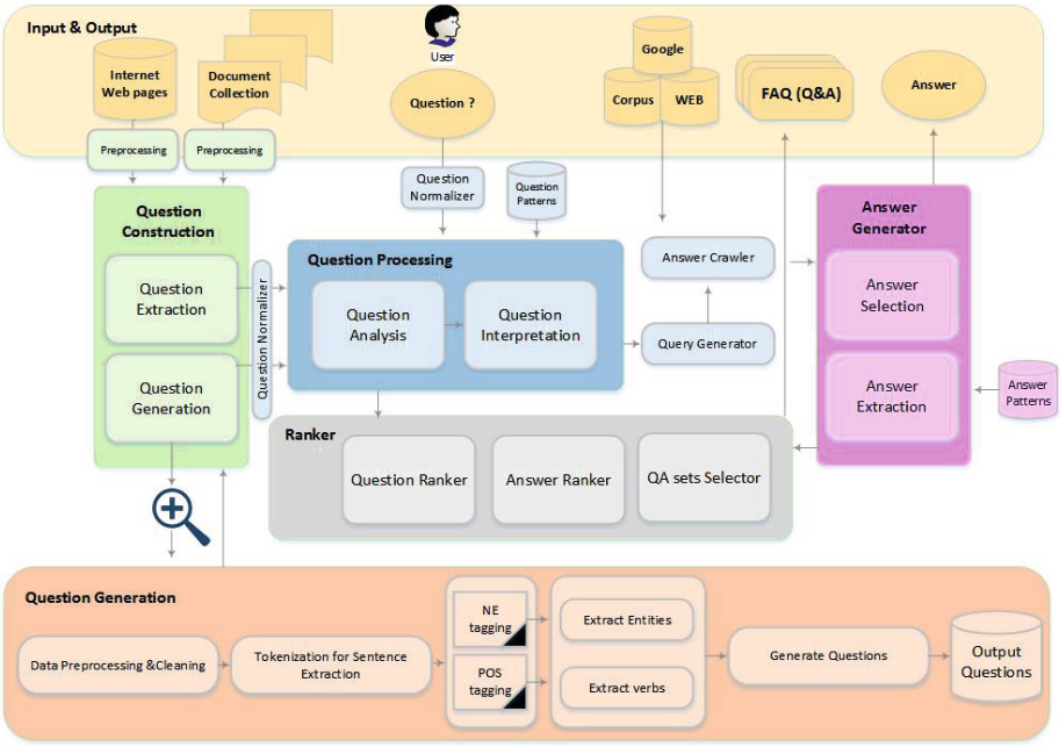


Figure 2.1: Auto FAQ Gen Architecture by Author of Auto-FAQ-Gen: Automatic Frequently Asked Questions Generation

The architecture presented was shown to be able to make Question and Answer Generation, Question Processing and a Ranker to rank answers.

2.1.1 Open Source Forums

(Hu et al., 2010) (Makino, Noro, & Iwakura, 2016) has made progress in selecting, weighing, clustering and ranking contextual keywords in order to achieve questions abstraction and ultimately enhance the process of finding questions and related answers in open source forums. The author then proposed the solution to be in a format of semi-automatic FAQ generation.

2.1.2 Neural Networks

Neural Networks are a collection of algorithms modelled after the human brain. It is a collection of algorithms used to identify patterns. It is able to learn independently by evaluating unstructured material with the capabilities of self-learning and adjusting weights to reach the optimal outcome.

Convolutional neural networks (CNN) and RNN has paved it's way towards text processing tasks for the past decade. In (Duan, Tang, Chen, & Zhou, 2017), CNN was proposed as a method to handle questions generation based on a given passage, it was done in a retrieval based format while RNN was propose to perform questions generation with a generation-based method. Both neural network were able to pull off question pattern prediction, question topic selection, question ranking and lastly question generation. The proposed model is shown to be able to outperform the traditional methods such as MAP or MRR.

2.1.3 Context Matching

Another issue arises, where most of the time, the questions asked by the user might not be easily classified to fit with the existing questions in the FAQ database. This is because the questions asked by the user might be in a different form or context (Makino et al., 2016). For example, the question asked by the user might be in a different language, or the question asked by the user might be in a different form.

Grammatically, misspellings are another set of issue that we can forsee when it comes to building automated questions answerings, highlighted by (Kothari, Negi, Faruquie, Chakaravarthy, & Subramaniam, 2009).

Context Matching is important when it comes to ranking the answers to the user query. The traditional way to score similarity is naive, where the score is calculated by levenshtein distance. (S. Zhang, Hu, & Bian, 2017) However, this method is not effective when it comes to ranking the answers. This is because the levenshtein distance is not able to capture the semantic meaning of the words. Another concept comes to mind which is BOW, the bag of words model, (Zhou, Liu, Liu, Zeng, & Zhao, 2013) highlighted to us where, BOWs similarity matching algorithm can be used on processed text (stop words removal, stemming, etc) to calculate the similarity between the query and the answer, however similarly to levenshtein distance, BOWs is not able to capture the semantic meaning of the words, which will bring false conceptual similarity between the query and the answer.

Word knowledge or Word Embedding is one of the solution to the traditional

similarity matching algorithm. (Zhou et al., 2013) proposed a method that uses word knowledge to improve the similarity matching algorithm. The model stitch together a knowledge base to a word, which then constructs a knowledge table which consists of the raw words, Hypernyms, Synonyms and Associative concept of words. This solution breaks the traditional similarity matching algorithm, where the similarity is calculated by the semantic meaning of the words, instead of the raw words. (Othman, Faiz, & Smaïli, 2019) has made progress on proposing a method called "WEKOS (Word Embedding, Kmeans and COSine based method)". WEKOS (Word Embedding, Kmeans and COSine) is a new method for question retrieval. The word embeddings of a question are weighted and averaged to get an overall representation of the question. The continuous word representations are learned in advance using the continuous bag-of-words (CBOW) model. (Othman et al., 2019) The cosine similarity is used to calculate the similarity between the average of the word vectors corresponding to the question and that of each existing question.

2.1.4 Rankings

Deep Matching Network also known as (DMN) is a deep learning model that generates a matching scores based of 2 matrices inputs. The matrices are made of the dot product of embeddings of every word of question with every word of answers.

Multi-hop Attention Network on the other hand is proven to be effective for reasonings tasks like question answering, which is our focus in this paper (Gupta & Carvalho, 2019). The network uses multiple "hops" of attention to gather information from a given input and make a prediction. It starts by encoding the input and then uses a decoder network that iteratively attends to different parts of the input, in order to generate an output.

In order to solve the issue of context misconception between the query and the faq. We can highlight the issue below.

query: I lost my credit card.
the question part of an FAQ: How do I request a new / replacement card?
the answer part of an FAQ: You can request a replacement card by signing in to the online banking and going to the information tab of your account.

Figure 2.2: Misconception

In the above example 2.2, the query only matched with the correct output with 2 characters "I" and "Card", that way, the resulting score will be very low, even though the FAQ is relevant to the query. To counter this issue, the authors of (Makino et al., 2016) proposed a method that predicts if the query corresponds to a FAQ using doc-

ument classifier. The paper begins to highlight that the proposed document classifier uses binary classifiers to classifier each faq to the query. The sentences are broken into unigrams, bigrams to learn the dependency relation between the query and the faq.

2.1.5 Semantic Networks

Automatically Generated Semantic Networks are networks that are generated by machine learning algorithms based on text or other forms of data. These networks are used to represent the relationships and connections between different concepts, entities, or topics in a structured manner, in order to facilitate various NLP or information retrieval tasks, such as classification, summarization, or question answering. They can be thought of as knowledge graphs constructed automatically from text.

Traditionally, QA (Question Answering) are managed using syntatic, semantic methods. However these methods are not pratical to perform well on low resource languages like Swahili. Semantic networks can be used to langauges like Swahili, where the language generally follows a subject-verb-object structure. (WANJAWA & MUCHEMI, 2020).

2.1.6 Topic Modeling

Topic Modeling is another great way to find Question and Answers Pairings. Topic modeling is a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. It is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. Latent Dirichlet Allocation (LDA) is a popular topic modeling technique that is used to discover topics in a collection of documents. LDA is a generative probabilistic model that assumes each document is a mixture of topics, and each topic is a mixture of words. The model is trained on a corpus of documents, and the topics discovered by the model can be used to label new, unseen documents. (Henß et al., 2012).

An architecture brought up by (Henß et al., 2012) highlights the pipeline on leveraging topic modelling to perform prediction on the question and answer pairings. The pipeline proposed extracts the topics from the corpus, then the topics are used to label the questions and answers, which then extracts answers that closely relates to the topic mined by the model.

2.2 Stop Words

Stop words are words that are filtered out before or after processing of natural language data (text). Though "stop words" usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools. The stopwords used in technical languages differs from the general stopwords list used by general application such as the NLTK library (Gerlach et al., 2019) (Sarica & Luo, 2020).

The properties of stopwords can be defined as follows (Ladani & Desai, 2020):

- Words used for connecting important words.
- Words that are not important to the meaning of the sentence.
- Words that occurs frequently in the text.

While it might be wise to perform stopwords removal tasks on tasks such as Information Retrieval (IR), classification, it is not advisable to perform stopwords removal on tasks like summarization. (Ladani & Desai, 2020).

(Gerlach et al., 2019) addresses the gap in identifying stopwords for texts in engineering fields which are not covered by the general stopwords list. The concluded stopwords list is statistically identified and evaluated by experts.

Phrases can be detected by using the algorithm of Mikolov at al (Mikolov,

Sutskever, Chen, Corrado, & Dean, 2013). The algorithm works by finding words that are frequently used together.

$$score(wi, wj) = ((count(wi * wj) - S)|N|/count(wi) * count(wj))$$

Term frequency-inverse-document-frequency (TFIDF) is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Many information retrieval systems use TFIDF as a central tool.

TF (Term Frequency) is the number of times a word appears in a document, divided by the total number of words in the document.

$$TF = \frac{Numberoftimestermtappearsinadocument}{Totalnumberoftermsinthedocument}$$

IDF (Inverse Document Frequency) is the log of the number of the documents in the corpus divided by the number of documents that contain the word w.

$$IDF = log_e(\frac{Totalnumberofdocuments}{Numberofdocumentswithtermtinit})$$

Term frequency-inverse-document-frequency (TFIDF) is then the product of

TF and IDF.

$$TFIDF = TF * IDF$$

This method particularly favours words that appear many times in a document. Both of this methods are used to score and rank words in a document, and potentially form a stopwords lists. An interesting fact is that with using the two methods mentioned above, the stopwords list can be generated for a specific domain.

2.3 Sentiment Analysis

Sentiment analysis is the process of determining whether a piece of writing is positive, negative, or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker. A common use case of sentiment analysis is to discover how people feel about a particular topic. For example, sentiment analysis can be used to discover how people feel about a new movie, or to learn about the general sentiment towards a new product.

While most sentiment analysis work are based of current social medias such as Twitter, Facebook, and Instagram, sentiment analysis can be applied to any text. For instance, (Alqaryouti, Siyam, Monem, & Shaalan, 2019) uses sentiment analysis to further investigate and understand the customers needs and wants for goverment entities.

2.3.1 Grananularity

Sentiment analysis can be performed on different levels of granularity. The level of granularity refers to the level of detail at which the sentiment is expressed. There are three levels of granularity: sentence-level, document-level, and aspect-level. Taking into account the level of granularity is important because it affects the type of sentiment analysis that can be performed, and the type of results that can be obtained (Alqaryouti et al., 2019).

2.3.2 Aspect Based

Aspect-Based Sentiment Analysis (ABSA) is a subfield of sentiment analysis that focuses on identifying and extracting opinions towards specific aspects or features of an entity such as a product, a service, or a person. It involves analyzing a piece of text to determine the sentiment expressed towards a particular aspect, rather than the overall sentiment of the text. This type of analysis is useful in understanding the specific opinions and attitudes of customers and users towards a particular product or service, and can be used in a variety of applications such as customer service, market research, and product development (Alqaryouti et al., 2019) (Liu, Chatterjee, Zhou, Lu, & Abusorrah, 2020) (Xue & Li, 2018) (M.Abdelgwad, A Soliman, I.Taloba, & Farghaly, 2022).

The goal of aspect-based sentiment analysis is to identify the sentiment expressed towards a particular aspect of an entity. For example, consider the following sentence:

"The food at this restaurant is great, but the service is terrible."

In this sentence, the overall sentiment might be positive, but in actuality the sentiment towards the food is positive, while the sentiment towards the service is negative. Therefore by using Aspect-based sentiment analysis can be used to identify the sentiment towards the food and the service separately. (Alqaryouti et al., 2019) took this intuition and applied it to the government sector. They used the ABSA to identify the sentiment towards the government services and the government entities.

In our research, we'll be using the ABSA model in the hopes where it'll improve the FAQ Ranker in terms of ranking their sentiment to potentially indicate the quality of the answer.

2.3.3 Traditional Methods

Lexicon and Corpus based approaches are unsupervised learning methods to understand the sentiment of a text. The lexicon based approach uses a predefined list of words coupled with the annotated polarity values of the word's sentiment. This approach has a major downside where there might be numerous words and expressions that are not included in the lexicons.

Corpus based approach however focuses more in predicting the word to be a prefixer or suffixer of the sentiment word. This approach is more accurate than the lexicon based approach, but it is still not perfect. (Syam Mohan E, 2021)

2.3.4 Machine Learning and Deep Learning

Multiple researches has proven that machine learning can make promising results in terms of ABSA specifically polarity classification. It was highlighted that attention-based or non attention based LSTM models can be used to achieve promising results (Syam Mohan E, 2021) (Liu et al., 2020)

SVM had received the highest popularity in terms of polarity classification, it was often used with association rules and feature selection such as PCA to achieve promising results (Syam Mohan E, 2021) (Zainuddin, Selamat, & Ibrahim, 2018) (Al-Smadi,

Qawasmeh, Al-Ayyoub, Jararweh, & Gupta, 2017)

Convolutional Neural Network were also used to predict polarity of a sentence, there were a handfull of attempts in using CNN coupled with Word2Vec that had been proven to be effective (Syam Mohan E, 2021) (Kumar, Pannu, & Malhi, 2020) (Rezaeinia, Rahmani, Ghodsi, & Veisi, 2019) (B. Zhang et al., 2019)

To learn long term associations and dependencies between texts in a sentence or a document. Long Short Term Memory (LSTM) was a popular choice in learning information dependencies across the history of sentences in a document. Multiple atempts had been made to take advantage of LSTM models in buildling polarity classification models (He, Lee, Ng, & Dahlmeier, 2018) (Ma, Zeng, Peng, Fortino, & Zhang, 2019) (Sindhu et al., 2019)

2.4 Summarization

Text summarization is the process of automatically generating a shorter version of a text that preserves the most important information. There are two main types of text summarization: extractive and abstractive. Extractive summarization involves selecting and concatenating important sentences from the original text, while abstractive summarization involves generating new sentences that summarize the meaning of the original text (Sharma & Sharma, 2022) (Nallapati, Zhou, Dos Santos, Gulcehre, & Xiang, 2016).

In contrast to general text summarization, summarizing software engineering related texts differs itself from the general text summarization. The main difference is that the software engineering related texts are often more technical and contain more technical terms, not to mention that at times, processing raw source codes can be a challenge (Moreno & Marcus, 2018).

The evaluation metrics used generally is to calcualate the compression rate denoted by the ratio of the length of the summary to the length of the original text.(Sharma & Sharma, 2022) The formula is as follows:

$$CompressionRate = \frac{Lengthof Summary}{Lengthof OriginalText}$$

2.4.1 Extractive Summarization

Extractive summarization involves selecting and concatenating important sentences from the original text. The most common approach to extractive summarization is to use a sentence ranking model to rank sentences in the original text, and then select the top ranked sentences to form the summary. The sentence ranking model can be trained using a variety of features, such as the number of named entities in the sentence, the number of words in the sentence which closely relates to TFIDF, the number of words in the sentence that are also in the original text, and the number of words in the sentence that are also in the summary (Sharma & Sharma, 2022).

CN-Summ is one of the notable extractive summarization model that was introduced by (Antiqueira, Oliveira, da F. Costa, & Nunes, 2009). CN-Summ is a neural network based model that uses a graph-like architecture to connect sentences that shared common significant nouns. The model consist preprocesses, a graph construction, and finally selecting the first x sentences as the summarization.

Similarly to using LDA in sentiment analysis (Syam Mohan E, 2021), LDA has also shed light in the field of extractive summarization. (Mashechkin, Petrovskiy, Popov, & Tsarev, 2011) proposed the use of LDA to extractive summarization. The model uses LDA to extract the most important topics in the original text, and then uses the topics to rank sentences in the original text.

2.4.2 Abstractive Summarization

Abstractive summarization involves generating new sentences that summarize the meaning of the original text. The most common approach to abstractive summarization is to use a sequence to sequence neural network to generate the summary. A sequence to sequence neural network such as RNN, LSTM is used because they are well-suited for processing sequences of data, such as text. The recurrent connections in RNNs allow them to maintain a hidden state that can capture information from previous steps in the sequence, which is useful for keeping track of contextual information as the model reads the input text. This allows the model to generate a summary that is both concise and semantically relevant to the input text (Sharma & Sharma, 2022).

AMR (Abstract Meaning Representation) was one of the RNN based methods that was introduced by (Banarescu et al., 2013). AMR is a neural network based model produces a single graph to represent time series information in the text to form an abstractive summary.

ATSDL is another attempt where the paper introduce a use of combining advantages of extractive and abstractive summazy to form a hybrid model. The model uses a sequence to sequence neural network to generate the summary, and then uses a sentence ranking model to rank the sentences in the summary. The top ranked sentences are then selected to form the final summary (Sharma & Sharma, 2022). A phrase collection model called MOSP was introduced to generate and learn phrase representations and their relationships in a document, the model solves the problem of unusual terms, which abstrative models faced most of the time.

2.5 Evaluation Metrics

Evaluating the success rate of extracting FAQ can be subjective, not all datasets coincide with the same domain. Therefore, a manual evaluation approach are mostly used to evaluate the success rate of a question finding process (Jijkoun & de Rijke, 2005). Manual classification of the questions pairs are carried out and the results are compared with the results of the question finding process.

However there are semi-automated frameworks that can potentially be used to evaluate the success rate of a question finding process. Alternatively, a user feedback system can be constructed, where a user can provide feedback on the accuracy of the question finding process. The feedback can be used to evaluate the success rate of the question finding process (Raazaghi, 2015).

CHAPTER 3

METHODOLOGY

3.1 Introduction to the framework

The following structure, which comprises a series of operations to be done on the data, is what we offer as a solution to these problems in order to solve them. Figure 3.1 is an illustration of the entire structure. The data collector and the inference engine are the two basic components that make up the framework. In the paragraphs that follow, I will dissect each part of the structure into its component phases. To make it simpler to keep track of where we are in the framework, we have allotted numbers to the several important stages (as indicated in the graphic below), and from this point on, we will refer to these labels instead of the descriptions.

Our whole pipeline includes all of the essential components and systems to comprehensively handle the problem. As a result of the modular design of the framework, it is feasible to replace any one of the individual stages with a different implementation if this proves to be necessary. Because of its scalability, the system may be applied to the management of extremely large datasets. The structure is designed to function regularly even if some processing steps experience errors. This is made possible by the framework's construction. A distributed design was used in the development of the framework so that it may be utilised on a broad range of computer systems. The framework's ability to readily include more procedures as they become

necessary is made possible by the modular nature it possesses.

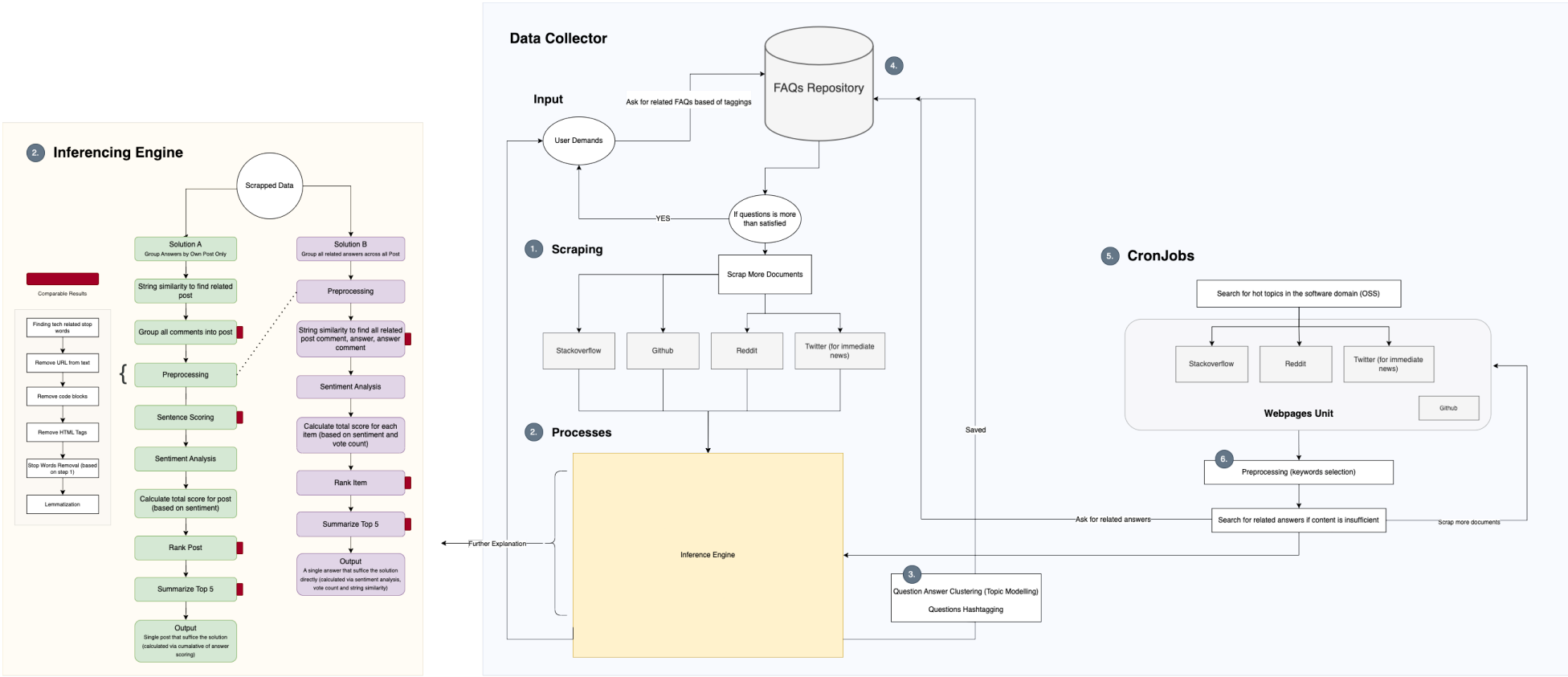


Figure 3.1: Framework

3.2 Introduction

In the following, we will go through everything that led us to the suggested framework. This will include a discussion of the ways in which the data are prepared, studied, and processed, as well as an evaluation of the approach that was developed as a consequence.

3.3 Data Collection (Scraping) – 1.0

To begin presenting the framework, we must first have access to the dataset; without it, we will be unable to provide proofs or draw conclusions about the suggested framework. For us, the dataset is vital since it determines the good or bad of the assessment process.

According to our suggested framework 3.1, we have four primary data sources for datapoints: Stack Overflow, GitHub (Issues), Reddit, and Twitter. Four of these provide us with data that is fairly distinct. The data on Stackoverflow is in the form of questions and answers, but the data on GitHub is in the form of code majority speaking. Reddit data is more likely to contain chained responses, whereas Twitter data is more likely to have rapid, real-time updates.

In terms of FYP1, Stackoverflow will be our primary data source because it makes it much easier for us to demonstrate whether or not our method works. As a result, we'll be collecting our information from Stack Overflow.

We utilised the stack overflow api to locate relevant topics on stackexchange

depending on the developer's query. We generate an API token on the Stack Overflow gateway since the API is not publicly available.

The Stack Exchange API is a RESTful API that allows you to access data from Stack Exchange sites. It is a read-only API, which means that you can only retrieve data from the sites, not modify it. The API is available at api.stackexchange.com/docs. You can use the API to retrieve data from Stack Exchange sites in a variety of formats, including JSON, XML, and JSONP. We'll be quering the data in JSON format.

Example of a query to the Stack Exchange API:


```
{
  "tags": [
  "node.js",
  "reactjs",
  "react-hooks"
  ],
  "owner": {
  "account_id": 26363344,
  "reputation": 9,
  "user_id": 20019220,
  "user_type": "registered",
  "profile_image": "https://lh3.googleusercontent.com/...",
  "display_name": "George Prethesh",
  "link":
      "https://stackoverflow.com/users/20019220/george-prethesh"
  },
  "is_answered": false,
  "view_count": 27,
  "answer_count": 3,
  "score": 0,
  "last_activity_date": 1674042867,
  "creation_date": 1674039763,
```

```
"question_id": 75158231,
"content_license": "CC BY-SA 4.0",
"link": "https://stackoverflow.com/questions/75158231/..",
"title": "React useEffect OnSubmit Rendering Post api multiple
    times"
},
```


The accompanying screenshot displays the response from the Stack Exchange API call. As you can see, it provides us with a number of useful pieces of data, including the tags, owner, title, link, and many more besides, however in this instance we are just concerned with the tags and the link. We may use the link to open a new browser tab and copy and paste the information from the website.

To do so automatically without human inteference, we will need to resort to the scraping technique in order to get the necessary information from the relevant web sites. Because of this, selenium is essential.

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms. It is used to automate web applications for testing purposes, but is certainly not limited to just that. Bots that perform web scraping, data mining, load testing, network traffic recording, and screen scraping are all common uses for the tool. Selenium is a portable framework for testing and automating web applications. Selenium provides a playback tool for authoring tests without learning a test scripting language (Selenese). It also provides a test domain-specific

language (Selenium-IDE) to write tests without learning a general-purpose programming language. The tests can then run against most modern web browsers. Selenium runs on Windows, Linux, and Mac OS X. Selenium is open source software released under the Apache 2.0 license.

However, there is a problem since most websites nowadays use anti-bot procedures to block automated programmes. Since our scraper may be blocked from accessing a Cloudflare-protected website, this complicates web scraping.

Therefore, we need to combine selenium with the ultrafunkamsterdam created and maintained python package undetected chromedriver to properly exploit its potential. This bundle is a selenium wrapper that enables us to utilise selenium without being noticed by the website. The website will deny us access if it discovers that we are using selenium, so knowing this is crucial.

When all of that is complete and in place, we can proceed to scrape all of the posts. The python code is designed to be error fail save, with the bulk of the code wrapped inside try catches blocks, so that if an error does occur, it will not stop the current process. This is critical since we'll be extracting more than 200-300 posts most of the time.

The list of our items of interest is as follows:

- Post id

- Post link
- Post Title
- Post Body
- Post date
- Post Votecounts
- Comment id
- Comment score
- Comment username
- Comment text
- Comment Date Time
- Answer id
- Answer Text
- Answer Body
- Answer Date Time
- Answer Votecounts
- Answer Comment id
- Answer Comment Text
- Answer Comment Body

- Answer Comment Date time
- Answer Comment Votecounts

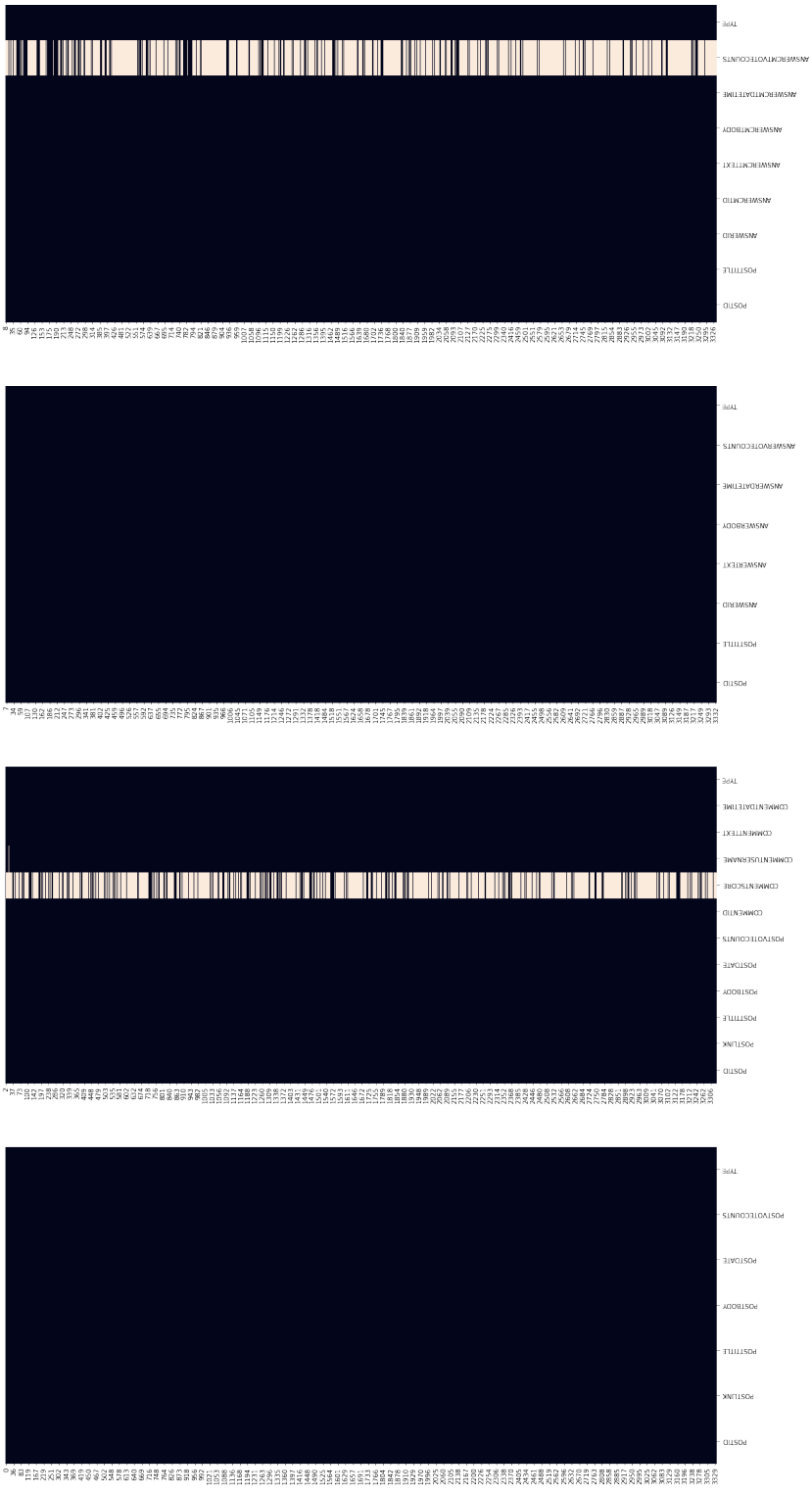
Because Selenium has a capability called "Find Element," it is quite easy for us to zero in on all of the components that are of interest. The information is structured in a manner similar to that of a dictionary, with the post id acting as the key and a list of all objects of interest performing the function of the value. After that, the data with the modifications are saved as a csv file. It is important to take note that the CSV file has a separate row for each comment, response, and response comment. This is due to the fact that we want to do a more in-depth review of the data.

We are going to scrape articles that include the word "React UseEffect" in order to acquire the necessary data. As a direct result of this, we now have a.csv file that has 3336 rows and 22 columns.

Notably, the web scraper is programmed to run once per day in order to provide us with the most recent data possible. This is necessary since data is always shifting, and we aim to compile the most up-to-date information possible. In addition to this, the scraper will be housed on a server so that it may expand as the needs of the business dictate.

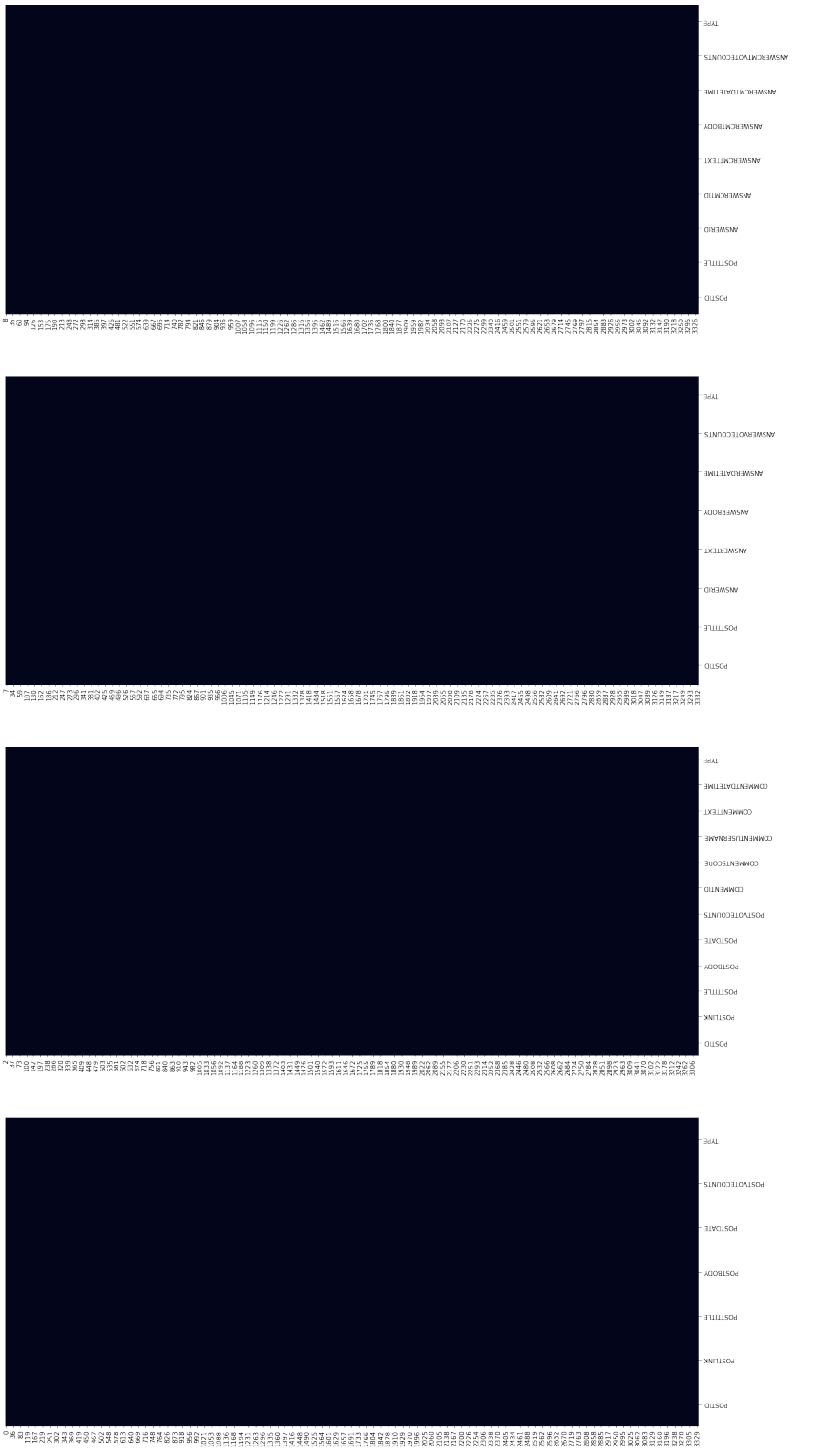
3.4 Data Cleaning

It is important to note that the data is not perfect. There are some null values that need to be addressed. The following are the null values that need to be addressed:



From the diagram we can see that everything is perfect except two columns

namely the Comment_Score and Answer_Comment_Score. The reason for this is that the comments and answer comments do not have a score. With that being said, we can replace the null values with 0. The following is the result of the replacement:

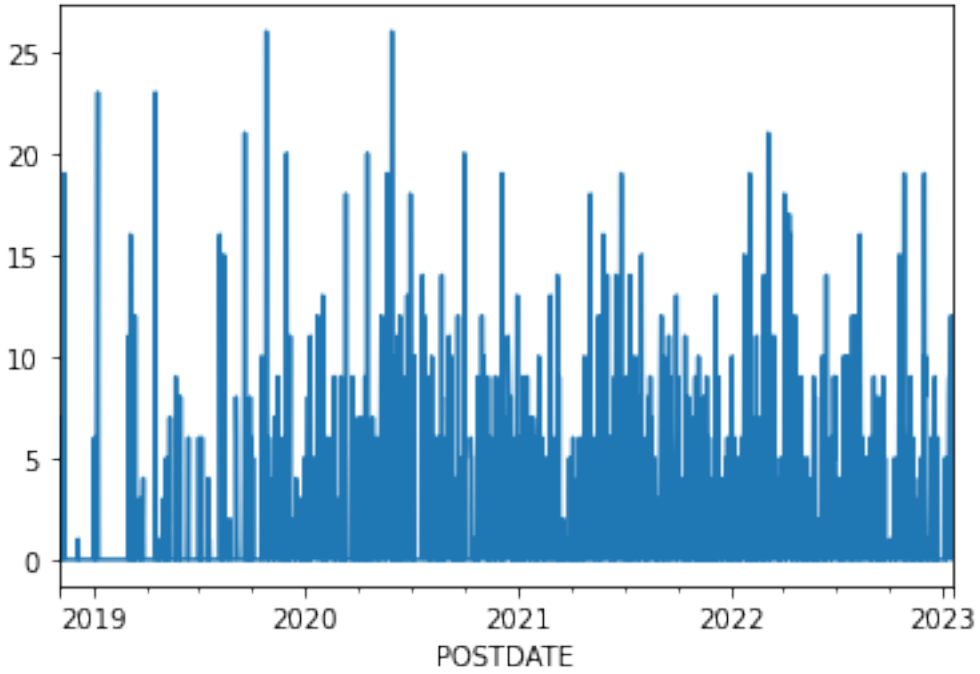


3.5 Data Exploration

It is usually a good idea to examine the data rather than just skim at it, so always do both.

First we should look at how many unique title there are. This is important since we want to make sure that we have enough data to work with. After analyzing the data, we can see that there are 660 unique titles. This is a good sign since it means that we have enough data to work with.

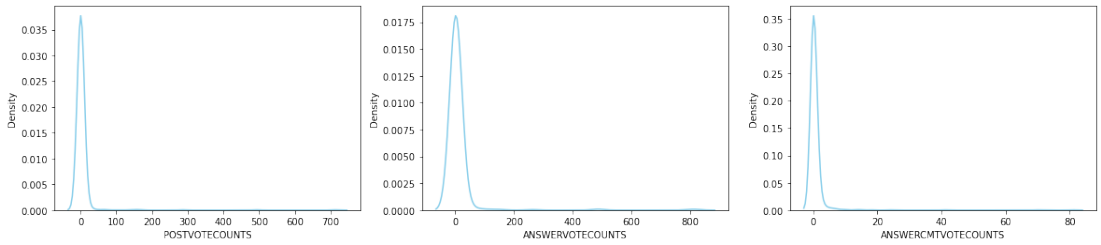
In order to analyze the longetivity of the current topic on stackoverflow, we will plot the time span distribution of all postings as a whole.



As you can see, the number of posts per day is pretty constant. This is a good

sign since it means that the number of people using the React UseEffect is still there and therefore it might gives us a better result.

Next, we will look at the votecounts distribution across all of the dataset as a whole, it is important to note that the votecounts is the number of votes that a post has received, the higher the votecounts, the more popular the post is. We can probably inference that the more popular the post is, the more likely it is to be answered, and the more likely it is to be answered correctly.

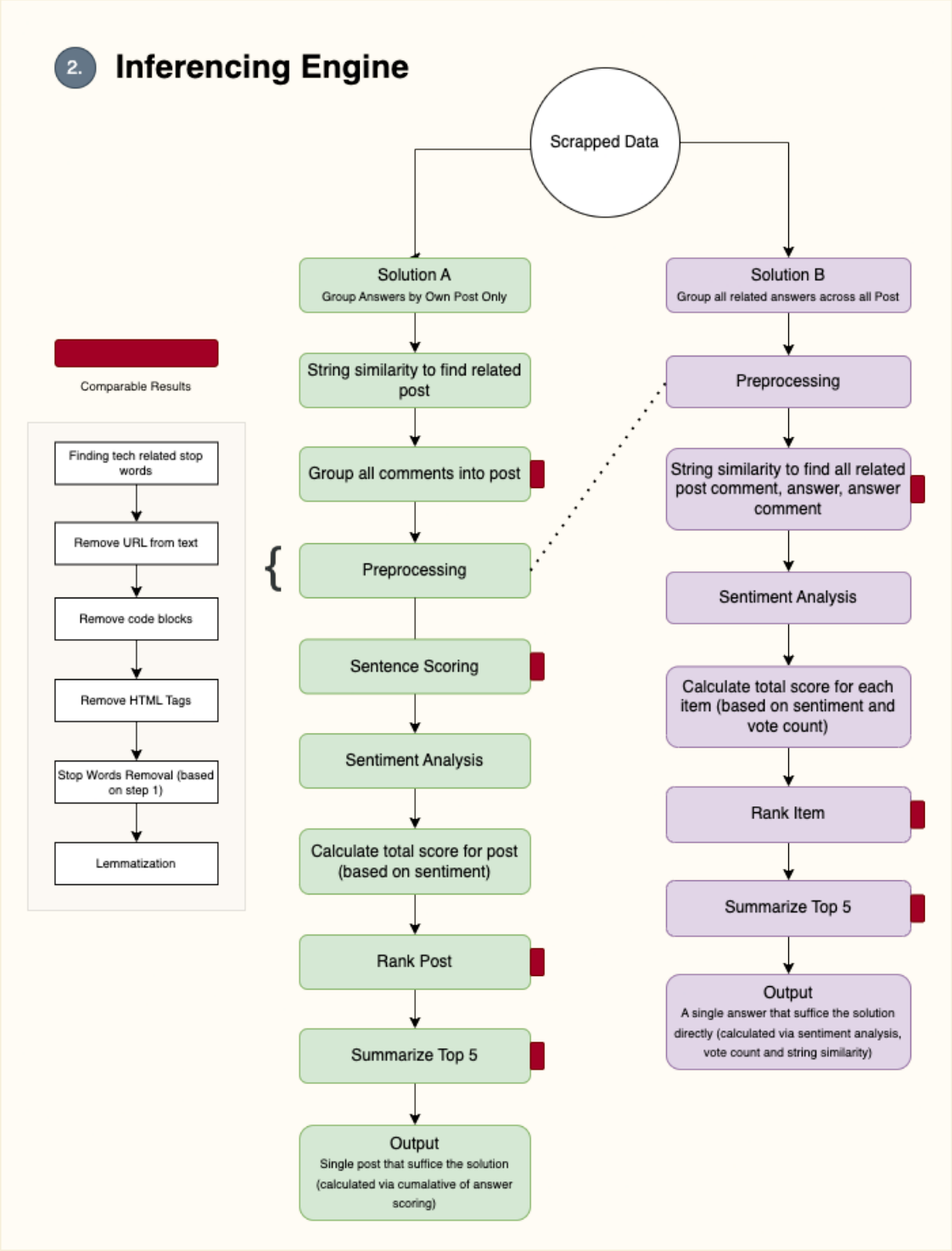


As you can see, the distribution is skewed to the left. This is not a good sign as it means that the majority of the posts have a low votecounts. With that being said, this motivates us to explore more data to be mined from the dataset and facilitate to our work.

3.6 Inferencing Engine – 2.0

We've attempt to propose an inference engine which will handle the core parts of the system. Everything from preprocessing, sentence scoring, and sentence selection will be handled by the inference engine. With the below diagram, it is a close up view of the inference engine. We'll be going through each of the components and

explain what they do.



In the proposed inference engine system, we came up with 2 ideas to solve

the problem, the one highlighted in green in color is solution A meanwhile the one highlighted in purple is solution B. In FYP 1, we'll be only implementing solution A, but we'll be explaining both of them in detail and the pros and cons of each of them.

3.6.1 Solution A

Posts titles are usually short and concise, and they are usually the first thing that people see when they are looking for a solution to their problem. Therefore, it is intuitive to assume that the title of the post tells us what the post is about. With that being said, this proves that all comments and answers should be incapsulate in the post itself, and the post should be view as a whole when we perform ranking. This is the idea behind solution A. Essentially, we're ranking the posts incorporating the comments and answers into the post itself.

3.6.2 Solution A - Architecture

A close up view of the architecture of solution A can be seen in the following figure 3.2.

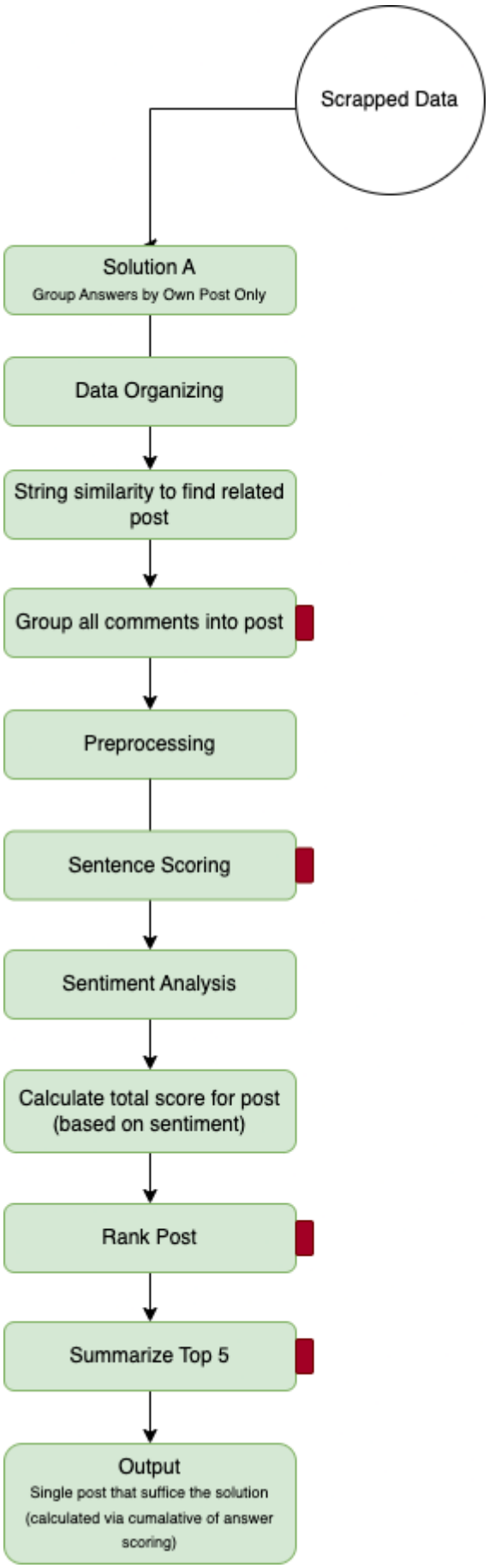


Figure 3.2: Solution A Architecture

3.6.2 (a) Data Organizing

First, we need to sort the information into the appropriate buckets. Posts, comments, answers and answers comments are the four primary types of information we separate apart. The goal here is to facilitate our work with the data. The reason for this is that the data in the csv is not organized in a way that is conducive to our work, we can see that in the following diagram:

So essentially we went from this:

```
All Columns DF

Index(['POSTID', 'POSTLINK', 'POSTTITLE', 'POSTBODY', 'POSTDATE',
       'POSTVOTECOUNTS', 'COMMENTID', 'COMMENTSCORE', 'COMMENTUSERNAME',
       'COMMENTTEXT', 'COMMENTDATETIME', 'ANSWERID', 'ANSWERTEXT',
       'ANSWERBODY', 'ANSWERDATETIME', 'ANSWERVOTECOUNTS', 'ANSWERCMTID',
       'ANSWERCMTTEXT', 'ANSWERCMTBODY', 'ANSWERCMTDATETIME',
       'ANSWERCMTVOTECOUNTS', 'TYPE'],
      dtype='object')
```

To this:

```
Post DF
Index(['POSTID', 'POSTLINK', 'POSTTITLE', 'POSTBODY', 'POSTDATE',
       'POSTVOTECOUNTS', 'TYPE'],
      dtype='object')
```

```
Post Comment DF
Index(['POSTID', 'POSTLINK', 'POSTTITLE', 'POSTBODY', 'POSTDATE',
       'POSTVOTECOUNTS', 'COMMENTID', 'COMMENTSCORE', 'COMMENTUSERNAME',
       'COMMENTTEXT', 'COMMENTDATETIME', 'TYPE'],
      dtype='object')
```

```
Answer DF
Index(['POSTID', 'POSTTITLE', 'ANSWERID', 'ANSWERTEXT', 'ANSWERBODY',
       'ANSWERDATETIME', 'ANSWERVOTECOUNTS', 'TYPE'],
      dtype='object')
```

```
Answer Comment DF
Index(['POSTID', 'POSTTITLE', 'ANSWERID', 'ANSWERCMTID', 'ANSWERCMTTEXT',
       'ANSWERCMTBODY', 'ANSWERCMTDATETIME', 'ANSWERCMTVOTECOUNTS', 'TYPE'],
      dtype='object')
```

3.6.2 (b) Matching Posts based on String Similarity

The first step is to match the posts based on string similarity. This is done by using the python package fuzzywuzzy. Fuzzy Wuzzy is a Python library for doing approximate and partial string matching using Levenshtein distance.

There are two methods we've used in fuzzy wuzzy, the first one is the partial ratio method, which is used to compare the similarity of two strings. fuzz.partial_ratio(str1, str2) method compares two strings and returns a score between 0 and 100, indicating the similarity of the two strings. The score is based on the longest contiguous matching substring between the two input strings.

fuzz.token_sort_ratio(str1, str2) method also compares two strings and returns a score between 0 and 100, indicating the similarity of the two strings. The score is based on the number of matching tokens (i.e., words) between the two input strings, after sorting the tokens alphabetically in both input strings.

In short, partial_ratio checks for longest matching substrings and token_sort_ratio checks for matching tokens after sorting them alphabetically.

The reason for this is to better match the posts with the developer requirements.

But we did not stop there, Fuzzy Wuzzy is entirely based on levenshtein distance, which is a metric for measuring the difference between two sequences. The levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. This might introduce some issues as the levenshtein distance is not perfect. For example, the levenshtein distance between "react" and "reactjs" is 2, which is not ideal.

Another string similarity method is then proposed, namely spaCy, which is a free, open-source library for advanced Natural Language Processing (NLP) in Python. It is designed specifically for production use and helps to build applications that process and “understand” large volumes of text.

The notable differences between FuzzyWuzzy and spaCy are:

- FuzzyWuzzy is based on Levenshtein distance, which is a metric for measuring the difference between two sequences. The levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other.
- spaCy is based on word embeddings, which is a type of word representation that allows words with similar meaning to have a similar representation. Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. It is a distributed representation for the text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems.

- FuzzyWuzzy is faster than spaCy, but spaCy is more accurate.
- FuzzyWuzzy is more suitable for matching short strings, while spaCy is more suitable for matching long strings.

To further filter out the results, we only take the results that have a score of 50 and above, as the results with a score below 50 are not relevant to the developer requirements, and we do not want to waste our time on them.

To test the accuracy of the two methods, we've prompted a random query – **"Useeffect hook rerenders infinitely"** , and we've compared the results of the two methods. Just for context there are 660 unique posts in the dataset.

The result is very contradictory, as the result of FuzzyWuzzy gives us a whopping 112 matches, while the result of spaCy gives us 58 matches. But if we take a closer look, we can see that the result of FuzzyWuzzy is more accurate, as the result of Spacy contains a lot of false positives, some of the matches are even unrelated to the developer requirements.

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	77	79	78.0
0	64651759	react useEffect hook causes infinite loop	71	71	71.0
0	58557877	React useEffect hook infinity loop	62	75	68.5
0	65558836	React useEffect hook is causing infinite loop	69	68	68.5
0	71835956	React UseEffect hook Issue Rendering an Interval	66	70	68.0
...	...	...	...	...	...
0	73334298	React useEffect every 5 seconds with setInterv...	51	50	50.5
0	60524845	React useEffect and clearInterval	45	56	50.5
0	59725069	React useEffect hook register callback with ac...	54	47	50.5
0	70258999	React useEffect() infinite re-render for getti...	46	55	50.5
0	61127662	React useEffect restarting timer	47	54	50.5

112 rows × 5 columns

Figure 3.3: Fuzzy Wuzzy Results on the query "Useeffect hook rerenders infinitely"

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.4: Spacy Results on the query "Useeffect hook rerenders infinitely"

However more testings needed to be taken place to truly crown the winner. Hence, we move forward to test the capabilities of both models using another query prompt, this time around we went for a longer sentence – **"How to solve useEffect hook rerenders infinitely?"**

This time around, the result is more or less on the same ratio, as the result of FuzzyWuzzy gives us 132 matches, while the result of spaCy gives us 195 matches.

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	72	67	69.5
0	64651759	react useEffect hook causes infinite loop	72	63	67.5
0	58557877	React useEffect hook infinity loop	68	66	67.0
0	62568196	React useEffect not rerendering	77	56	66.5
0	64917867	How to sto React useEffect hook from rendering...	71	60	65.5
...	...	...	...	...	...
0	56831803	Prevent infinite loop when updating state via ...	45	56	50.5
0	63540163	React useEffect: Why is my value undefined ini...	55	46	50.5
0	67658202	React useEffect syntax reason on mount	50	51	50.5
0	58830374	React useEffect for Loading Data	53	48	50.5
0	61433211	React useEffect hook loop, dependency problem	58	43	50.5

132 rows × 5 columns

```
132 number of titles found
```

Figure 3.5: Fuzzy Wuzzy Results on the query "How to solve useEffect hook rerenders infinitely?"

	postId	title	score
0	65404350	How to correctly add event listener to React u...	0.902468
0	70710122	How to use setInterval with react useEffect ho...	0.899317
0	72632981	How to fulfill React useEffect missing depende...	0.871420
0	74609238	How to stop the infinite loop inside this Reac...	0.861767
0	67343507	How to give a boolean indicator to React useEf...	0.833936
...	...	...	...
0	70790681	React useEffect dependency not triggering from...	0.507214
0	72784960	I dont understand why this infinite loop wont ...	0.505654
0	70110116	React useEffect does infinite loop	0.504187
0	71157226	Why is React useeffect not updated when props ...	0.502435
0	73369425	React useEffect invalid hook call	0.500079

195 rows × 3 columns

```
195 number of titles found
```

Figure 3.6: Spacy Results on the query "How to solve useEffect hook rerenders infinitely?"

At this point, it is clear that Spacy is more accurate than FuzzyWuzzy, but it

is also clear that FuzzyWuzzy is faster than Spacy. Hence, we decided to use both methods to filter out the results, and we will use the results that are common between the two methods.

3.6.2 (c) Group all answers, comments, and answer comments by post_id

The next step is to group all answers, comments, and answer comments by post_id, so this way, all answers, comments, and answer comments can be concatinated into a single dict with the post_id as the key, and thus all answers, comments, and answer comments can be viewed as a single string. Viewing everything as a single string is crucial for us as it's way easier to do sentence scording, sentiment analysis and other NLP tasks on a single string than on a list of strings. For that to happen we create a class called **GroupedComments** which is responsible for grouping all answers, comments, and answer comments by post_id.

```
class GroupedComments:

  def ___init___(self, title, post, post_comments, answers,

      answer_comments):

       self.title = title

       self.post = post

       self.post_comments = post_comments

       self.answers = answers

       self.answer_comments = answer_comments
```

We then use a simple python script that iterates over the data and groups all an-

swers, comments, and answer comments by post_id. The disctionary can be visualized as follows:

```
"post_id": {
  "title": "title",
  "post": "post",
  "post_comments": "post_comments",
  "answers": "answers",
  "answer_comments": "answer_comments"
}
```

Listing 3.1: Grouped Comments Dictionary Example

Since now that the data has been formatted nicely it is very easy for us to track the context of the data and to do further analysis on the data.

Rank Post - 1st Round

With all of the comments grouped together we can begin our first analysis on the data. The context of the ranking this round is that, all of the comments, answers, answers comments are grouped under the individual parent posts, and the posts are ranked by the similarity score between the query and the post.

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	77	79	78.0
0	64651759	react useEffect hook causes infinite loop	71	71	71.0
0	58557877	React useEffect hook infinity loop	62	75	68.5
0	65558836	React useEffect hook is causing infinite loop	69	68	68.5
0	71835956	React UseEffect hook Issue Rendering an Interval	66	70	68.0
...	...	...	...	...	...
0	73334298	React useEffect every 5 seconds with setInterv...	51	50	50.5
0	60524845	React useEffect and clearInterval	45	56	50.5
0	59725069	React useEffect hook register callback with ac...	54	47	50.5
0	70258999	React useEffect() infinite re-render for getti...	46	55	50.5
0	61127662	React useEffect restarting timer	47	54	50.5

112 rows × 5 columns

Figure 3.7: Rank Post - 1st Round

3.6.2 (d) Preprocessing - Finding tech related stop words

3.6.2 (d) The next step is to find tech related stop words. This is done in order to remove the stop words from the data. Stop words are words that are not important to the data, and they are usually words that are used frequently in the data.

However, finding stop words is a very tedious task, in the previous yeaers, most of the time we would use manually curated stopwords lists to remove stop words from the data (Gerlach et al., 2019),the stop words list can come from a variety of sources, such as a list of stop words from a specific domain, or a list of stop words from a specific language. In our case, we're dealing with a very specific domain, which is the engineering domain, normal stop words lists are not suitable for our case, as they are not specific to the engineering domain. If we continue to use normal stop words lists, we will end up with noises in our datasets.

Stop words list can be found by using a wordcloud to find the most frequent words in the data, and then removing the most frequent words from the data. However, this method is not very accurate, as it does not take into account the context of the data, and it does not take into account the importance of the words in the data.

(Sarica & Luo, 2020) has made tremendous effort on curating the stopword list for the engineering domain, they've done it by analyzing the data found in patent documents which mostly describes domains related to engineering. With that being said, we decided to use their stopword list to remove stop words from the data.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.8: Comparison of the results before and after removing stop words

3.6.2 (e) Remove Special Characters

The next step is to remove special characters from the data. Since we're scrapping the data from StackOverflow directly with all sorts of special characters, the data is not clean at all. Having special characters in the dataset might causes a lot of unwanted trouble, therefore we decided to remove all special characters from the data. We've constructed a table 3.1 to show all the special characters that we've found in the data.

Table 3.1: Special Character

Case	VoteCount	Dates	All Columns
1	(	, License: CC BY-SA 4.0	segFault
2	)	(	
3	,	)	
5	’	,	

We’ve used the string replace method to remove all special characters from the data.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.9: Comparison of the results before and after removing stop words

3.6.2 (f) Preprocessing - Extract URL from text

It is essential to incorporate URLs into the data, as these may be mined for information on the context of the data. We are able to effortlessly extract the URLs from the data as it is scraped since the data contains HTML tags. The BeautifulSoup package was utilised so that we could access the URLs included inside the data. As a consequence of this, the url can be stored in a separate column of the dataset so that it can be referred to in the future.

In FYP 2, we are going to make an effort to put in place a system that can automatically determine the context relationships between retrieved URLs and extract more information from the URLs that have been obtained. Applying a score system is one way to determine whether or not the URL in question is relevant to the data. If the scoring method wasn't implemented, the process of the system going to each of the urls would take a very significant amount of time.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.10: Comparison of the results before and after extracting URLs

3.6.2 (g) Preprocessing - Remove Punctuation

The next step is to remove punctuation from the data. Punctuation is usually surrounded by “““ or ““““ in the data. We’ve used the string replace method to remove all punctuation from the data.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.11: Comparison of the results before and after removing punctuation

3.6.2 (h) Preprocessing - Remove code blocks

The next step is to remove code blocks from the data. Code blocks are usually surrounded by “““ or “““ in the data. We’ve used the string replace method to remove all code blocks from the data.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.12: Comparison of the results before and after removing code blocks

3.6.2 (i) Preprocessing - Remove HTML Tags

The next step is to remove HTML tags from the data. HTML tags are usually surrounded by < and > in the data. Again similarly to how we process code blocks, we've used the string replace method to remove all HTML tags from the data.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.13: Comparison of the results before and after removing HMTL Tags

3.6.2 (j) Preprocessing - Stop Words Removal

In the previous step 3.6.2 (d), we've found a list of stop words that are related to the technology domain. The removal is not done on the previous step because the text might contain noisy data Therefore, we decided to remove the stop words after the data is cleaned. We've used the string replace method to remove all stop words from the data.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337

0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.14: Comparison of the results before and after removing stop words

3.6.2 (k) Preprocessing - Lemmatization/Stemming

The process of truncating a word to it's root or base unit is called stemming or lemmatization.

In the discipline of Natural Language Processing, text normalisation techniques like stemming and lemmatization are employed to get sentences, words, and docu-

ments ready for analysis. For instance, the terms "kick" and "kicked" are both forms of the verb "to kick," thus it's possible that you'll want your natural language processing application to recognise this. This is the idea of stripping down many uses of a term to its essential meaning.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337
0	70110116	React useEffect does infinite loop	0.549249
0	65261753	React useEffect hook with empty dependency ren...	0.548529
0	72751014	React useEffect infinite loop cause	0.548313
0	66709882	How does react useEffect work with useState hook?	0.548105
0	70053657	Unexpected behaviour using React useEffect hook	0.546131
0	61849911	React useEffect hook missing dependencies lint...	0.542206
0	67250636	How do I fire React useEffect hook only once a...	0.532225
0	73138128	Can I use an if statement within a React useEf...	0.532120
0	62066729	React useEffect hook is not working with rails	0.530559
0	56097549	Loading spinner with react useEffect hook and ...	0.527682
0	70710122	How to use setInterval with react useEffect ho...	0.526649
0	66493554	Why is my React useEffect not loading again wh...	0.525649
0	74609238	How to stop the infinite loop inside this Reac...	0.525113
0	60296644	React useEffect hook does not fire when prop d...	0.520637
0	73065012	react useEffect async triggering eachother pro...	0.516272
0	73038997	React UseEffect hook only runs twice even with...	0.514029
0	53253940	Make React useEffect hook not run on initial r...	0.504022
0	70189522	React useEffect hook and eslint warning	0.500513

58 number of titles found

Figure 3.15: Comparison of the results with using stemming or lemmatization

Techniques like lemmatization or stemming can be achieved by using the famous Python NLTK package.

However, whether to utilise stemming or lemmatization is a contentious issue. Because stemming is a primitive heuristic procedure that cuts off the ends of words in the aim of reaching this objective properly most of the time, it often involves the removal of derivational affixes.

Lemmatization, on the other hand, is a more complex technique that takes into consideration morphological examination of the words. It does this by the use of a lexicon and morphological analysis of words, with the goal of removing only inflectional ends and returning the base or dictionary form of a word, known as the lemma.

To truly examine each of it's effectiveness on our dataset, we've made a comparison on which one is better. The result is shown below.

To conclude, we've decided to use lemmatization instead of stemming because we want to keep the words as much as possible for the summarization process.

3.6.2 (l) Sentence Scoring

At this stage, all of our sentences should be clean and ready to be scored. The scoring process is done by using the TF-IDF algorithm. The TF-IDF algorithm is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. The algorithm is composed of two terms: the first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document; the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.

The TF-IDF score is the product of these two terms. The TF-IDF score increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general.

The TF and IDF scores are defined as follows:

$$\text{TF} = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in the document}} \tag{3.1}$$

$$\text{IDF} = \log \frac{\text{Total number of documents}}{\text{Number of documents with term t in it}} \tag{3.2}$$

At last, The TF-IDF scoring can be simplified into a single formula as shown below:

$$\text{TF-IDF} = \text{TF} \times \text{IDF} \tag{3.3}$$

To apply the TF-IDF algorithm, we first join all the sentences into a single string. Then, we use the TF-IDF algorithm to calculate the TF-IDF score for each sentence and average it, thus the score will be tied with the post. The result is shown below.

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337

Figure 3.16: Post scoring using sentence scoring

Rank Post - 2nd Round

After the sentence scoring process, we've got the score for each post. The next step is to rank the post based on the score. With that being said, an analysis and comparison can be done to compare the result of the first round and the second round. The result is shown below.

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	77	79	78.0
0	64651759	react useEffect hook causes infinite loop	71	71	71.0
0	58557877	React useEffect hook infinity loop	62	75	68.5
0	65558836	React useEffect hook is causing infinite loop	69	68	68.5
0	71835956	React UseEffect hook Issue Rendering an Interval	66	70	68.0
...	...	...	...	...	...
0	73334298	React useEffect every 5 seconds with setInterv...	51	50	50.5
0	60524845	React useEffect and clearInterval	45	56	50.5
0	59725069	React useEffect hook register callback with ac...	54	47	50.5
0	70258999	React useEffect() infinite re-render for getti...	46	55	50.5
0	61127662	React useEffect restarting timer	47	54	50.5

112 rows × 5 columns

Figure 3.17: Rank Post - 2nd Round

3.6.2 (m) Sentiment Analysis

Sentiment analysis is the process of determining whether a piece of writing is positive, negative, or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker. A common use case of sentiment analysis is to discover how people feel about a particular topic.

Sentiment analysis is usually performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs. It can also be used to gauge the sentiment of the general public to a particular event.

But how does sentiment analysis take place in our framework?

For context, each posts are scored based on the TF-IDF algorithm. The score is then used to rank the post. However the score itself is not telling enough to determine whether the post is useful or not. We've made an assumption that the post with a positive sentiment is more likely to be useful than the post with a negative sentiment. Therefore, we decided to use the sentiment analysis to determine the sentiment of the post. Therefore, we decided to use the sentiment analysis to add another layer of filtering ont he ranking aspect of the framework.

3.6.2 (n) Sentiment Analysis – Model

The model we've used is the famous twitter roberta base sentiment analysis model, which has been used over 2 million times this month. It is a RoBERTa base

model trained on approximately 124M tweets from January 2018 to December 2021, and finetuned for sentiment analysis with the TweetEval benchmark. The model is able to predict and inference a probability score for each of the 3 classes: positive, negative, and neutral.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337

Figure 3.18: Results using sentiment analysis

3.6.2 (o) Rank Post

After the sentiment analysis process, we've got the sentiment for each post. The next step is to rank the post based on the sentiment. This will be the third round of ranking. With the figures below, we can see the comparison between the first round, second round, and third round of ranking.

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	77	79	78.0
0	64651759	react useEffect hook causes infinite loop	71	71	71.0
0	58557877	React useEffect hook infinity loop	62	75	68.5
0	65558836	React useEffect hook is causing infinite loop	69	68	68.5
0	71835956	React UseEffect hook Issue Rendering an Interval	66	70	68.0
...	...	...	...	...	...
0	73334298	React useEffect every 5 seconds with setInterv...	51	50	50.5
0	60524845	React useEffect and clearInterval	45	56	50.5
0	59725069	React useEffect hook register callback with ac...	54	47	50.5
0	70258999	React useEffect() infinite re-render for getti...	46	55	50.5
0	61127662	React useEffect restarting timer	47	54	50.5

112 rows × 5 columns

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	77	79	78.0
0	64651759	react useEffect hook causes infinite loop	71	71	71.0
0	58557877	React useEffect hook infinity loop	62	75	68.5
0	65558836	React useEffect hook is causing infinite loop	69	68	68.5
0	71835956	React UseEffect hook Issue Rendering an Interval	66	70	68.0
...	...	...	...	...	...
0	73334298	React useEffect every 5 seconds with setInterv...	51	50	50.5
0	60524845	React useEffect and clearInterval	45	56	50.5
0	59725069	React useEffect hook register callback with ac...	54	47	50.5
0	70258999	React useEffect() infinite re-render for getti...	46	55	50.5
0	61127662	React useEffect restarting timer	47	54	50.5

112 rows × 5 columns

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	77	79	78.0
0	64651759	react useEffect hook causes infinite loop	71	71	71.0
0	58557877	React useEffect hook infinity loop	62	75	68.5
0	65558836	React useEffect hook is causing infinite loop	69	68	68.5
0	71835956	React UseEffect hook Issue Rendering an Interval	66	70	68.0
...	...	...	...	...	...
0	73334298	React useEffect every 5 seconds with setInterv...	51	50	50.5
0	60524845	React useEffect and clearInterval	45	56	50.5
0	59725069	React useEffect hook register callback with ac...	54	47	50.5
0	70258999	React useEffect() infinite re-render for getti...	46	55	50.5
0	61127662	React useEffect restarting timer	47	54	50.5

112 rows × 5 columns

Figure 3.19: Rank Post - 3rd Round

3.6.2 (p) Summarization

At the final step, in order to fulfill a better user experience, the engine will then summarize the top 5 sentiment ranked posts.

Summarization is the process of shortening a text document with software, in order to create a summary with the major points of the original document. As the name suggests, it is a technique to reduce the size of the given text while trying to retain the most important information. A summarization is crucial in our framework as it will help the user to understand the post better, faster and easier.

3.6.2 (q) Summarization – Model

The model we've used is the very well known bart large cnn model, which has been used over 1 million times this month. BART model pre-trained on English language, and fine-tuned on CNN Daily Mail. It was introduced in the paper BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation

BART is a transformer encoder-encoder (seq2seq) model with a bidirectional (BERT-like) encoder and an autoregressive (GPT-like) decoder. BART is pre-trained by (1) corrupting text with an arbitrary noising function, and (2) learning a model to reconstruct the original text.

BART is particularly effective when fine-tuned for text generation (e.g. summarization, translation) but also works well for comprehension tasks (e.g. text classification, question answering). This particular checkpoint has been fine-tuned on CNN Daily Mail, a large collection of text-summary pairs.

Plot before and after here

	postId	title	score
0	58557877	React useEffect hook infinity loop	0.782165
0	64651759	react useEffect hook causes infinite loop	0.728888
0	60984978	React useeffect hook	0.696022
0	60112511	Setting hook state inside React useEffect hook	0.682884
0	63800700	react useEffect hook cleanup	0.673840
0	57772851	React useEffect hook load onsnapshot condition...	0.666406
0	71032390	React useEffect hook dependency array	0.648506
0	71357152	React useEffect hook running infinite loop des...	0.646135
0	56914826	Using react useEffect hook	0.642261
0	68008264	React useEffect fetch hook makes endless calls...	0.632083
0	65558836	React useEffect hook is causing infinite loop	0.621120
0	73369425	React useEffect invalid hook call	0.616057
0	70614166	React useEffect hook toggle issue	0.614819
0	68836817	Confusing React useEffect hook behaviour	0.606975
0	74111109	componentWillUnmount lifecycle with react usee...	0.605624
0	68910950	React useEffect hook doesn't clear interval	0.603321
0	68690569	How can I escape React useEffect infinite loop?	0.602124
0	55139386	componentWillUnmount with React useEffect hook	0.590641
0	68846124	React useEffect stop infinite loop	0.589314
0	59059431	React useeffect hook behaving not like i expected	0.587801
0	61433211	React useEffect hook loop, dependency problem	0.587443
0	63189274	How can I make this React useEffect hook work ...	0.578809
0	74379165	React UseEffect hook firing every time I click...	0.575062
0	59146735	React useEffect infinite loop	0.574421
0	67346406	react useEffect hook triggers only once althou...	0.572204
0	55088168	React useEffect hook not calling mocked function	0.571941
0	69074323	How can i check whether the object changes whe...	0.570510
0	59541356	React useEffect runs every time I scroll after...	0.570179
0	73227468	React useEffect hook will cause an infinite lo...	0.569777
0	75140656	How do I set a timer on React/UseEffect hook a...	0.567358
0	66506524	Refreshing Bootstrap4 Modal using React useEff...	0.566474
0	63688921	React useEffect hook issue	0.565556
0	65163988	React useEffect infinite loop despite empty array	0.563563
0	65673342	React useEffect hook does not call after recoi...	0.563404
0	71325566	How do I prevent unnecessary, repetitive side-...	0.562854
0	67030266	React useEffect hook event listener fired twice	0.562569
0	73534338	React UseEffect render infinite loop	0.561589
0	55011667	React useEffect hook referencing incorrect values	0.553651
0	66256470	React useEffect hook doesn't trigger when a de...	0.553495
0	64396705	React useEffect endless loop	0.551337

Figure 3.20: Results after summarization

3.6.2 (r) The Finale: The Final Results

We've come to the end of our framework using solution A. As you can see on the figure below, the result are promising, and the user is able to get the information they need with a very short text format. This helps the user to easily locate the posts that are useful to their query.

	postId	title	partial_ratio	token_sort_ratio	average
0	73534338	React UseEffect render infinite loop	77	79	78.0
0	64651759	react useEffect hook causes infinite loop	71	71	71.0
0	58557877	React useEffect hook infinity loop	62	75	68.5
0	65558836	React useEffect hook is causing infinite loop	69	68	68.5
0	71835956	React UseEffect hook Issue Rendering an Interval	66	70	68.0
...	...	...	...	...	...
0	73334298	React useEffect every 5 seconds with setInterv...	51	50	50.5
0	60524845	React useEffect and clearInterval	45	56	50.5
0	59725069	React useEffect hook register callback with ac...	54	47	50.5
0	70258999	React useEffect() infinite re-render for getti...	46	55	50.5
0	61127662	React useEffect restarting timer	47	54	50.5

112 rows × 5 columns

Figure 3.21: Rank Post - The Finale

3.6.3 Solution B

The first solution focuses on ranking posts, whereby all comments, answers and answers comments are grouped together with the post and is been seen as a whole entity. Closed loop context, is the issue that we'll be addressing in this solution. We'll be explaining the architecture, the issue of solution A in detail, and the approach that we'll be taking to solve the issue.

3.6.3 (a) Solution B - Issue of Solution A

The issue of solution A is the situation – closed loop context, meaning that the ranking of the posts solely depends on the numbers, interactivity and the sentiment of the comments, answers and answers comments of the post.

This is not a serious problem, however there is room for improvement. Jumping on another point of view, let's zoom out from the post context, let's view all of the comments, answers, and answer comments all together from every posts. The argument that we're trying to define here is, there is possibility that the answers from other posts are more relevant to the query than the answers from the post itself? The sole reason where the post that contains the better answers is not ranked is because the title of the post is not relevant to the query.

This is where the second solution comes in. We'll be using the same approach as solution A, however we'll be grouping all related answers across all posts and do string similarity, and the rest of the architecture will be more or less similar to solution A.

Note, this is a solution made up with an assumption of the possibility of the answers from other posts are more relevant to the query than the answers from the post itself. The actual implementation will be covered in FYP2.

3.6.3 (b) Solution B - Architecture

The architecture of solution B is very similar to solution A, the only difference is that we'll be grouping all related answers across all posts, we can see that in the figure 3.22

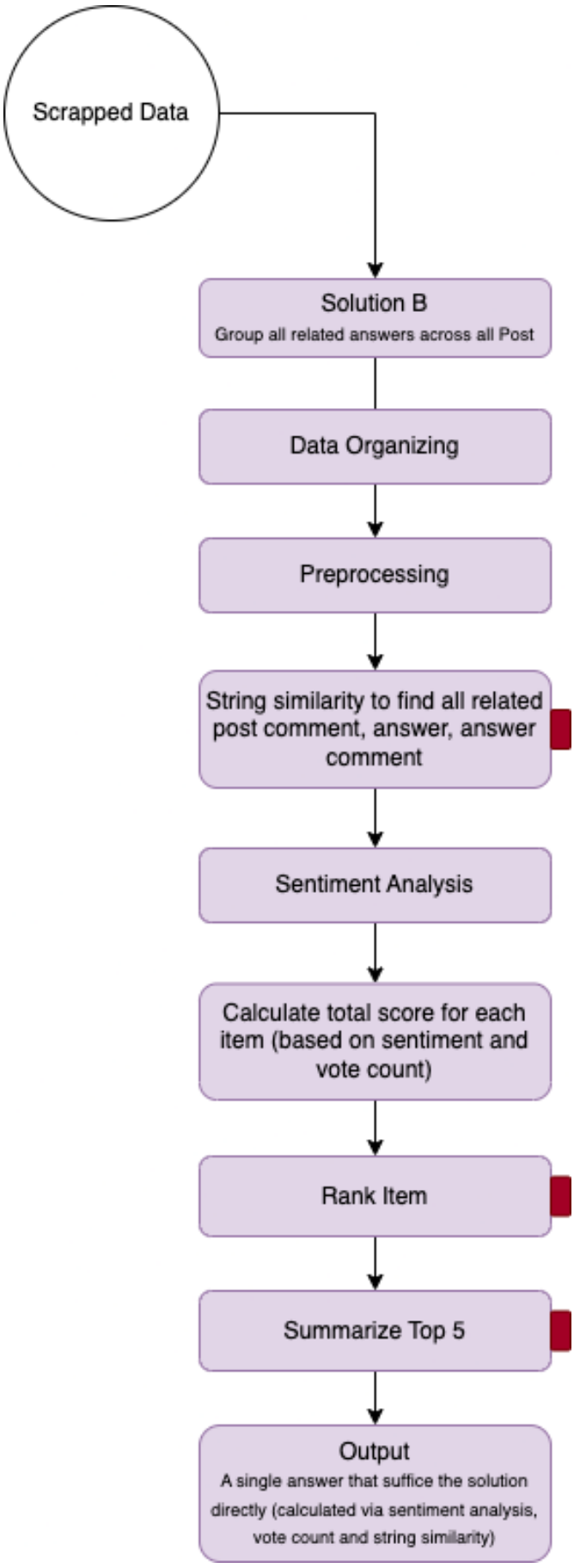


Figure 3.22: Solution B Architecture

3.6.3 (c) Solution B - Approach

The approach that we'll be taking is very similar to solution A, the only differentiating factor is that we're thinking on a wider context, we're not just thinking about the post itself, we're thinking about all the posts. All of the comments should be seen as equal no matter the parent post of the comments itself.

Data cleaning 3.4, organizing 3.6.2 (a), pre-processing 3.6.2 (d) will be done in the same way as solution A. Just that the data will not be grouped by post 3.6.2 (c), but rather by all posts.

This way all comments, answers, answers comments take a part in the ranking process. We believe that this is a better system as it excludes the bias of the post title, and it helps to rank the individual item that are more relevant to the query, eventually leading to a better user experience.

3.7 Topic Modeling

3.8 Real Time Capabilities Issue

One of the big issues for this framework to be generalized is the dataset. The dataset has to be generalized enough to be able to answer any query, that is simply impossible considering the wide range of the tehcnical field.

Real time scraping to populate the dataset is not a good idea as it will be very costly, time consuming, and cpu intensive, as our workers have to scale up to the

demand of the users.

Therefore, we've came up with two solutions to solve the problem, the first solution is to build a FAQs repository to store FAQs for further usage, and the second solution is to predict the next faq that will be asked.

3.9 FAQs Repository

With the first solution, a FAQs repository is needed to be built. The FAQs repository will be a collection of questions and answers that are related to the technical field. The way it works is that whenever a user asks a question, the system will check if the question is in the FAQs repository, if it is, the system will return the answer from the FAQs repository. If it's not, the system will then scrap all the related posts from stackoverflow, and do inferencing. The summarized result will be stored in the FAQs repository, and the user or anyone will be able to access the answer from the FAQs repository next time they ask the same question.

3.10 FAQs Repository - Architecture

In the below figure, we can see the architecture of the FAQs repository. The FAQs repository is a database that stores the questions and answers. The database is populated by the system, and the system will check the database whenever a user asks a question.

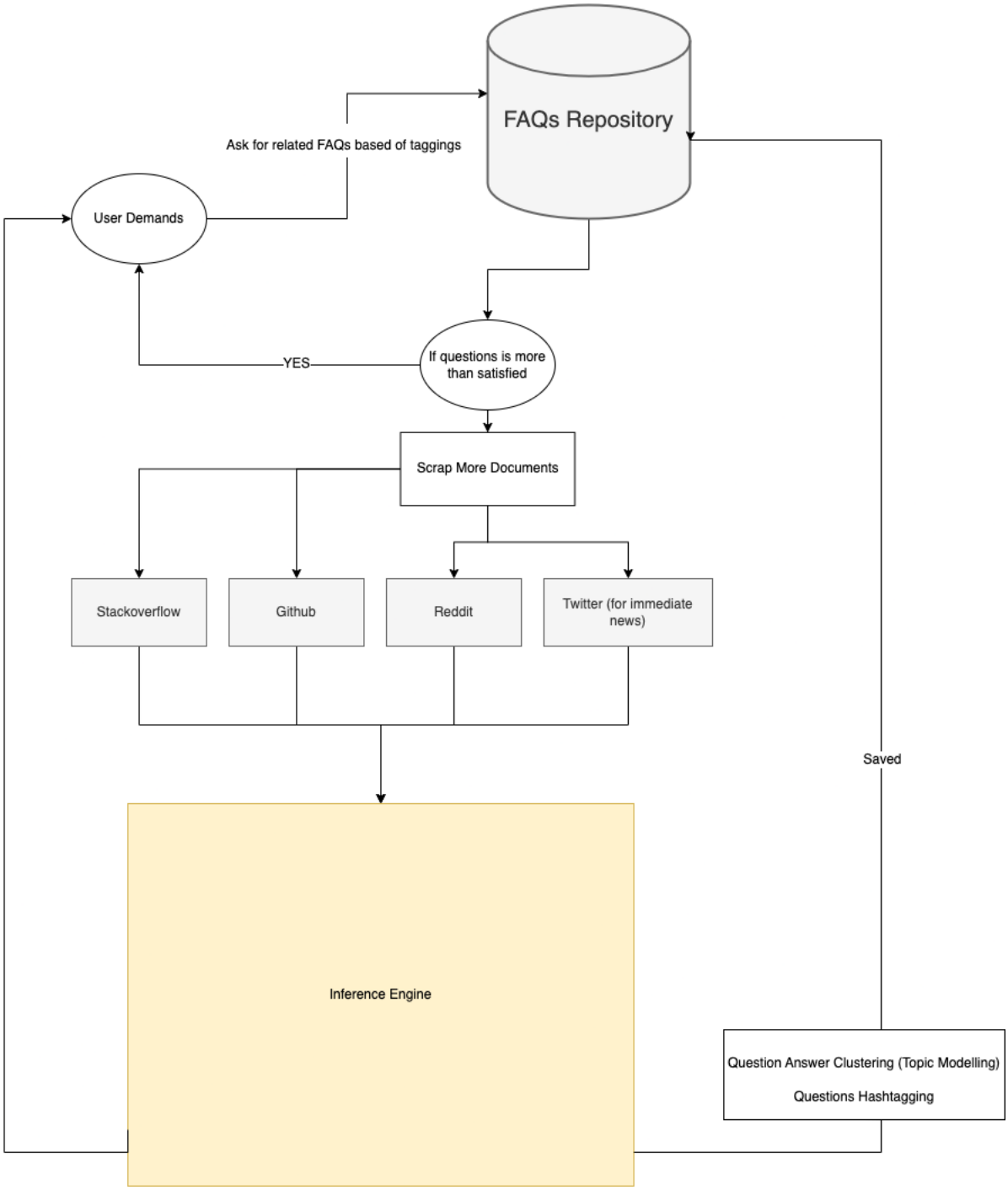


Figure 3.23: FAQ Repository Architecture

3.11 FAQs Repository - Approach

With the above architecture, the system will be functioning as stated below:

1. The user asks a question

2. The system will check if the question is in the FAQs repository

3. If the question is in the FAQs repository, the system will return the answer from the FAQs repository

4. If the question is not in the FAQs repository, the system will then scrap all the related posts from stackoverflow, and do inferencing

5. The summarized result will be stored in the FAQs repository, and the user or anyone will be able to access the answer from the FAQs repository next time they ask the same question.

With this architecture, we can ensure that the system is not only able to answer the question that is asked before, but also able to answer the question that is not asked before, also it can ensure that the system will not scrap the same question again and again, as it's already in the FAQs repository.

3.12 FAQs Repository - Database Design

The database of choice is MongoDB, as it's a NoSQL database, and it's very easy to scale up. As a plus point, MongoDB being a NoSQL database gives us the flexibility to store the data in anyway we wanted, and the structure of the data can be altered whenever.

The database will be structured as below:

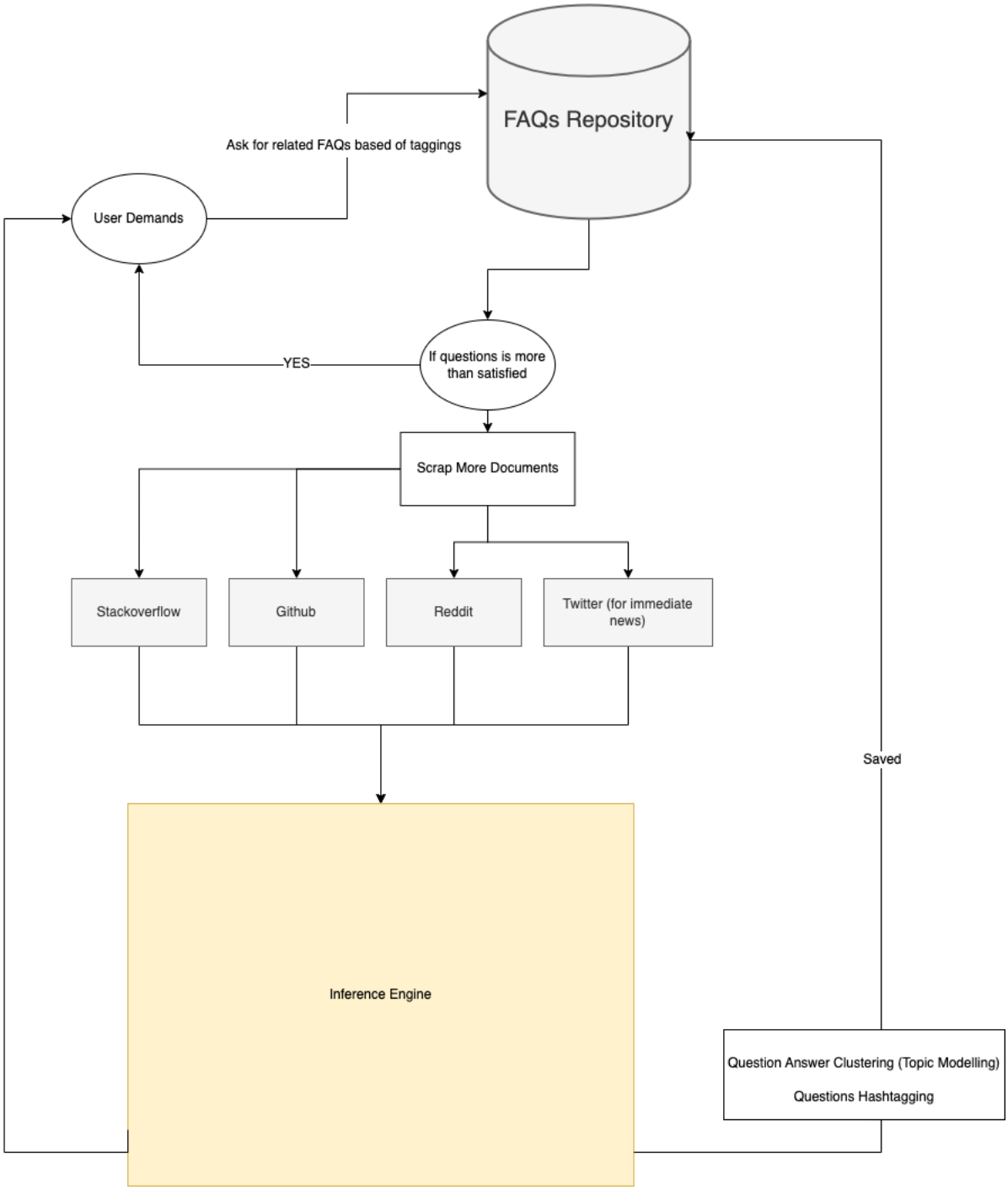


Figure 3.24: Database Design

3.13 FAQs Repository - System Design

An system design figure is shown below:

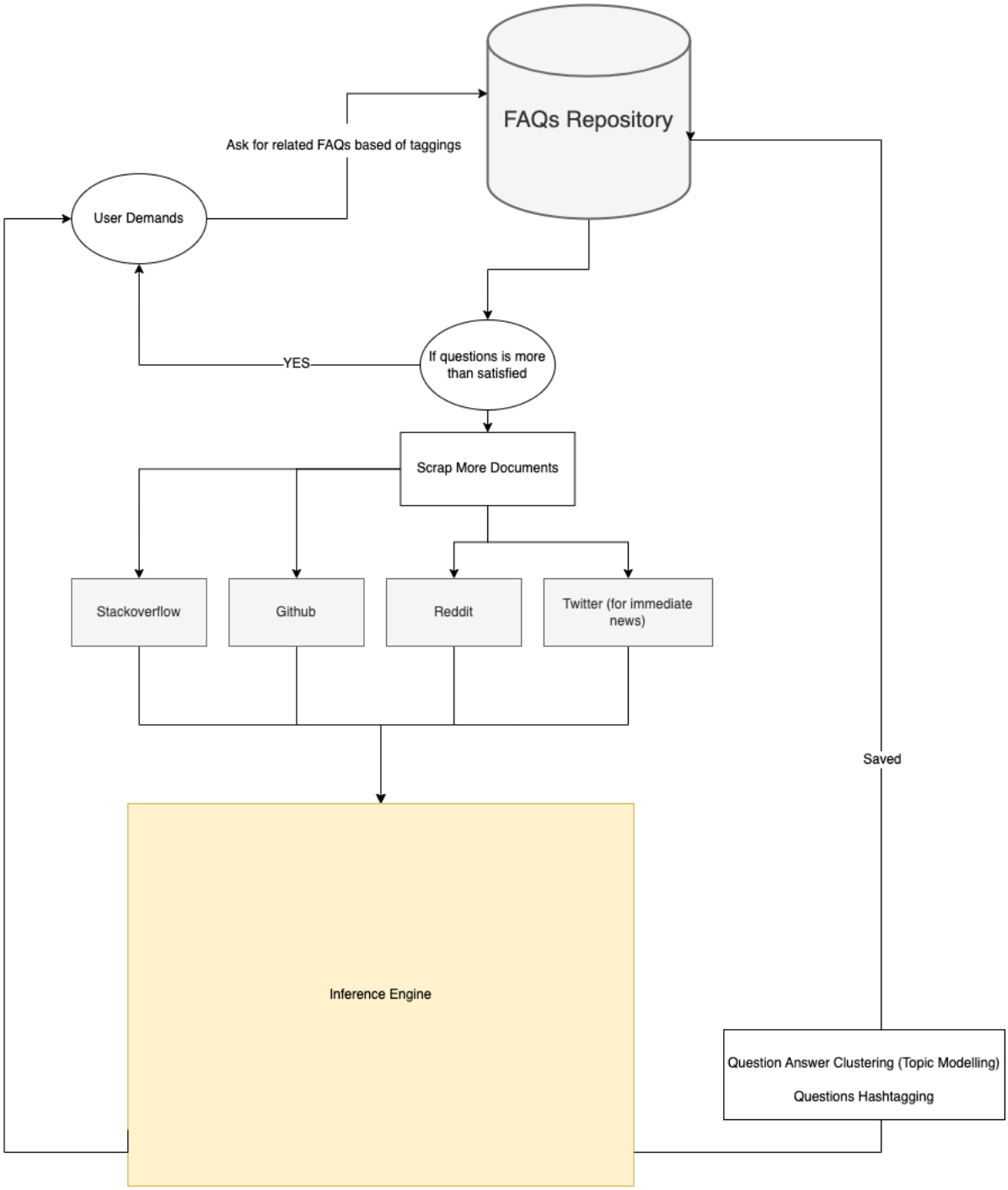


Figure 3.25: System Design

The entire backend will be built using NodeJs, as it's a very popular language, and it's very easy to scale. We'll be using a boilerplate created by the author of this

thesis, which consists of multi-threading and clustering capabilities. These factors will help us to scale up the system whenever we need to. The boilerplate is available at `https://github.com/Shaunmak1214/cluster-node-auth-sql-boilerplate.git`.

A few notable perks that the boilerplate has is that it has a built in authentication system, and it has a built in SQL database, and a whole lot more. All of the important features of the boilerplate will be listed below:

- Clustering in Node.js
- Server with Express.js
- Database schema and models using Sequelize ORM
- User authentication with JWT
- StandardJs for coding standards and styling
- Request validation using Express-validator
- Morgan and Winston for server side logging
- Swagger for API documentation

To host the backend, we'll be using AWS EC2, as it's the best cheap option, and it's very easy to scale according to user demands. Amazon Elastic Compute Cloud (Amazon EC2) offers the broadest and deepest compute platform, with over 500 instances and choice of the latest processor, storage, networking, operating system, and

purchase model to help you best match the needs of your workload. We are the first major cloud provider that supports Intel, AMD, and Arm processors, the only cloud with on-demand EC2 Mac instances, and the only cloud with 400 Gbps Ethernet networking. We offer the best price performance for machine learning training, as well as the lowest cost per inference instances in the cloud. More SAP, high performance computing (HPC), ML, and Windows workloads run on AWS than any other cloud.

Laslty to protect the servers from ddos attacks, we'll be using AWS WAF, which is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources. AWS WAF gives you control over which traffic to allow or block to your web applications by defining customizable web security rules. You can use AWS WAF to create security rules that block common attack patterns, such as SQL injection or cross-site scripting, and rules that filter specific types of traffic. AWS WAF is integrated with AWS Shield, a managed DDoS protection service that safeguards web applications running on AWS. AWS Shield Standard provides always-on detection and automatic inline mitigations that minimize application downtime and latency, and AWS Shield Advanced provides distributed denial of service (DDoS) attack protection, proactive engagement with AWS support, and usage metrics.

3.14 CronJobs - Predictive FAQ Generation System

While the first solution – FAQ Repository 3.9 is able to solve the problem, there's still one big problem. The FAQs repository solution works well if your question is already asked by someone, but what if your question is not asked by anyone before?

The system will not be able to answer your question, and you will have to wait for the lenghty process for the system to scrap, process, analyze, rank and summarize the data for you. This is not ideal, if you have tons of bugs to solve and you have to wait for the system to answer your question.

To solve this problem, we'll be using a cronjob to run the system every 24 hours, and the system will scrap, process, analyze, rank and summarize the data, and store it in the FAQs repository. This way, the system will be able to answer any question that is asked by the user, and the user will not have to wait for the system to answer their question.

One of the big question mark for this is, what to scrap?

For context, we have primarily 4 data sources, Stackoverflow, Reddit, Twitter and lastly Github.

3.15 CronJobs - Predictive FAQ Generation System - Architecture

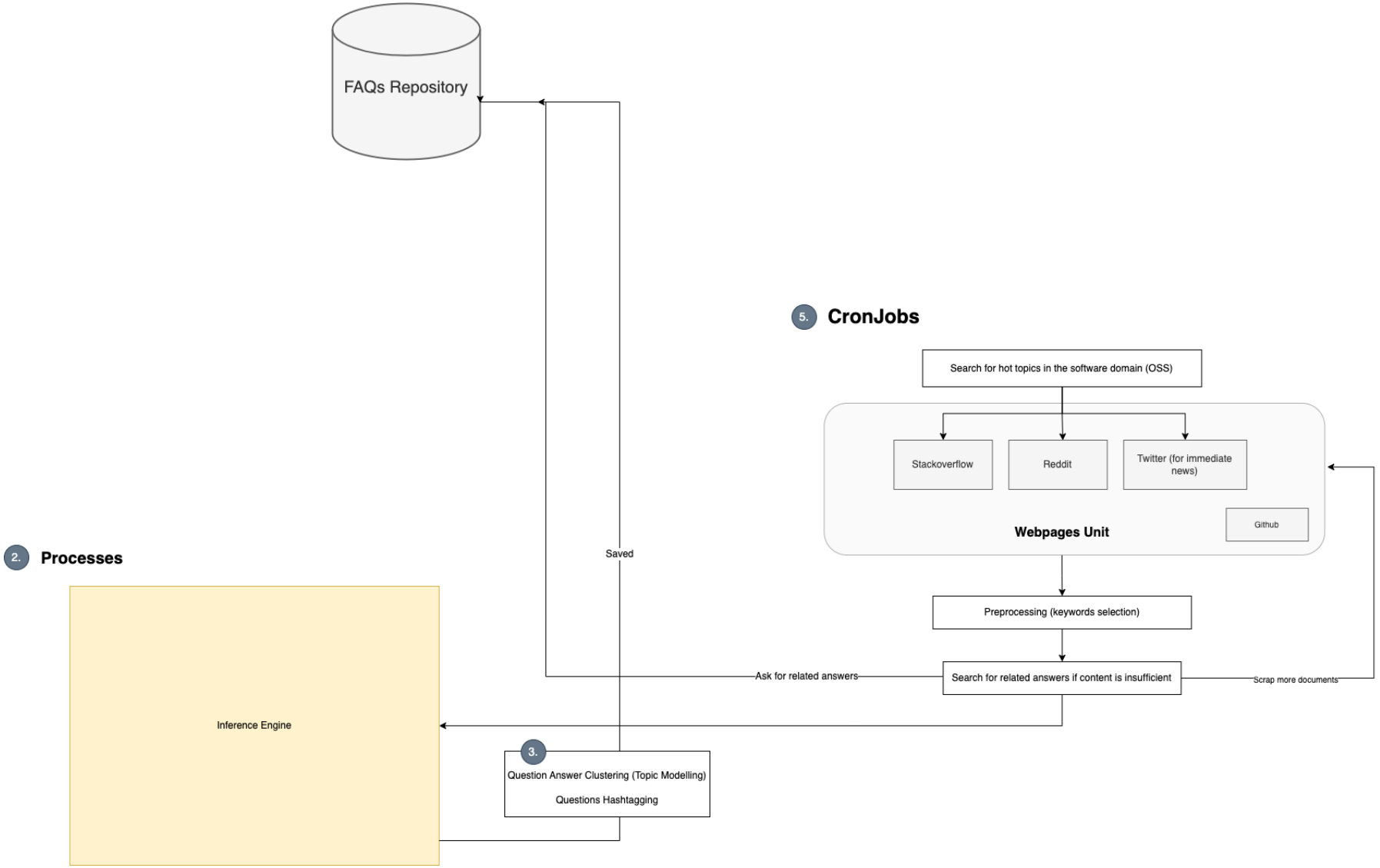


Figure 3.26: CronJobs Architecture

3.16 CronJobs - Predictive FAQ Generation System - Approach

With the above architecture, the system will be functioning as stated below:

1. The system will scrap all the posts from stackoverflow, reddit, twitter and github every 24 hours

2. Perform topic modeling on the scrapped data, Extracting keywords

3. If dataset is not sufficient (metrics will be explored), scrap more documents related to the topic.

4. Process, analyze, rank and summarize the data

5. Store the summarized data in the FAQs repository

Which the proposed architecture, we can improve the system where the system will populate itself with the most recent data, and in the hopes where the system is able to fit with the user's needs.

3.17 Result Analysis

3.18 Evaluation

3.19 Future Work

REFERENCES

[1] Alqaryouti, O., Siyam, N., Monem, A., & Shaalan, K. (2019, 11). Aspect-based sentiment analysis using smart government review data. *Applied Computing and Informatics*, *ahead-of-print*. doi: 10.1016/j.aci.2019.11.003

[2] Al-Smadi, M., Qawasmeh, O., Al-Ayyoub, M., Jararweh, Y., & Gupta, B. B. (2017). Deep recurrent neural network vs. support vector machine for aspect-based sentiment analysis of arabic hotels' reviews. *J. Comput. Sci.*, *27*, 386-393.

[3] Antiqueira, L., Oliveira, O., da F. Costa, L., & Nunes, M. (2009, 02). A complex network approach to text summarization. *Information Sciences*, *179*, 584-599. doi: 10.1016/j.ins.2008.10.032

[4] Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., ... Schneider, N. (2013, August). Abstract Meaning Representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse* (pp. 178–186). Sofia, Bulgaria: Association for Computational Linguistics. Retrieved from https://aclanthology.org/W13-2322

[5] Duan, N., Tang, D., Chen, P., & Zhou, M. (2017, September). Question generation for question answering. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 866–874). Copenhagen, Denmark: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D17-1090 doi: 10.18653/v1/D17-1090

[6] Gerlach, M., Shi, H., & Amaral, L. (2019, 12). A universal information theoretic approach to the identification of stopwords. In (Vol. 1). doi: 10.1038/s42256-019-0112-6

[7] Gupta, S., & Carvalho, V. R. (2019). Faq retrieval using attentive matching. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval* (p. 929–932). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3331184.3331294 doi: 10.1145/3331184.3331294

[8] He, R., Lee, W. S., Ng, H. T., & Dahlmeier, D. (2018, July). Exploiting document knowledge for aspect-level sentiment classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 2: Short papers)* (pp. 579–585). Melbourne, Australia: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P18-2092 doi: 10.18653/v1/P18-2092

[9] Henß, S., Monperrus, M., & Mezini, M. (2012). Semi-automatically extracting faqs to improve accessibility of software development knowledge. In *2012 34th international conference on software engineering (icse)* (p. 793-803). doi: 10.1109/ICSE.2012.6227139

[10] Hu, W.-C., Yu, D.-F., & Jiau, H. C. (2010). A faq finding process in open source project forums. In *2010 fifth international conference on software engineering advances* (p. 259-264). doi: 10.1109/ICSEA.2010.46

[11] Jijkoun, V., & de Rijke, M. (2005). Retrieving answers from frequently asked questions pages on the web. In *Proceedings of the 14th acm international conference on information and knowledge management* (p. 76–83). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1099554.1099571 doi: 10.1145/1099554.1099571

[12] Kothari, G., Negi, S., Faruquie, T., Chakaravarthy, V., & Subramaniam, L. (2009, 01). Sms based interface for faq retrieval. In (p. 852-860). doi: 10.3115/1690219.1690266

[13] Kumar, R., Pannu, H., & Malhi, A. (2020, 04). Aspect-based sentiment analysis using deep networks and stochastic optimization. *Neural Computing and Applications*, *32*. doi: 10.1007/s00521-019-04105-z

[14] Ladani, D. J., & Desai, N. P. (2020). Stopword identification and removal techniques on tc and ir applications: A survey. In *2020 6th international conference on advanced computing and communication systems (icaccs)* (p. 466-472). doi: 10.1109/ICACCS48705.2020.9074166

[15] Liu, H., Chatterjee, I., Zhou, M., Lu, X. S., & Abusorrah, A. (2020). Aspect-based sentiment anal-

ysis: A survey of deep learning methods. *IEEE Transactions on Computational Social Systems*, *7*(6), 1358-1375. doi: 10.1109/TCSS.2020.3033302

[16] Ma, X., Zeng, J., Peng, L., Fortino, G., & Zhang, Y. (2019, apr). Modeling multi-aspects within one opinionated sentence simultaneously for aspect-level sentiment analysis. *Future Gener. Comput. Syst.*, *93*(C), 304–311. Retrieved from https://doi.org/10.1016/j.future.2018.10.041 doi: 10.1016/j.future.2018.10.041

[17] M.Abdelgwad, M., A Soliman, T. H., I.Taloba, A., & Farghaly, M. F. (2022). Arabic aspect based sentiment analysis using bidirectional gru based models. *Journal of King Saud University - Computer and Information Sciences*, *34*(9), 6652-6662. Retrieved from https://www.sciencedirect.com/science/article/pii/S1319157821002482 doi: https://doi.org/10.1016/j.jksuci.2021.08.030

[18] Makino, T., Noro, T., & Iwakura, T. (2016). An faq search method using a document classifier trained with automatically generated training data. In R. Booth & M.-L. Zhang (Eds.), *Pricai 2016: Trends in artificial intelligence* (pp. 295–305). Cham: Springer International Publishing.

[19] Manes, S. S., & Baysal, O. (2019). How often and what stackoverflow posts do developers reference in their github projects? In *2019 ieee/acm 16th international conference on mining software repositories (msr)* (p. 235-239). doi: 10.1109/MSR.2019.00047

[20] Mashechkin, I., Petrovskiy, M., Popov, D., & Tsarev, D. (2011, 11). Automatic text summarization using latent semantic analysis. *Programming and Computer Software*, *37*, 299-305. doi: 10.1134/S0361768811060041

[21] Menéndez, H. D. (2021). Software testing or the bugs' nightmare..

[22] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in neural information processing systems*

(Vol. 26). Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[23] Moreno, L., & Marcus, A. (2018). Automatic software summarization: The state of the art. In *2018 ieee/acm 40th international conference on software engineering: Companion (icse-companion)* (p. 530-531).

[24] Nallapati, R., Zhou, B., Dos Santos, C., Gulcehre, C., & Xiang, B. (2016, 02). Abstractive text summarization using sequence-to-sequence rnns and beyond. In (p. 280-290). doi: 10.18653/v1/K16-1028

[25] Othman, N., Faiz, R., & Smaïli, K. (2019). Enhancing question retrieval in community question answering using word embeddings. *Procedia Computer Science*, *159*, 485-494. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050919313857 (Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 23rd International Conference KES2019) doi: https://doi.org/10.1016/j.procs.2019.09.203

[26] Raazaghi, F. (2015). Auto-faq-gen: Automatic frequently asked questions generation. In D. Barbosa & E. Milios (Eds.), *Advances in artificial intelligence* (pp. 334–337). Cham: Springer International Publishing.

[27] Razzaghi, F., Minaee, H., & Ghorbani, A. A. (2016). Context free frequently asked questions detection using machine learning techniques. In *2016 ieee/wic/acm international conference on web intelligence (wi)* (p. 558-561). doi: 10.1109/WI.2016.0095

[28] Rezaeinia, S. M., Rahmani, R., Ghodsi, A., & Veisi, H. (2019). Sentiment analysis based on improved pre-trained word embeddings. *Expert Syst. Appl.*, *117*, 139-147.

[29] Sarica, S., & Luo, J. (2020, 06). Stopwords in technical language processing..

[30] Sharma, G., & Sharma, D. (2022). Automatic text summarization methods: A comprehensive

review. *SN Computer Science*, *4*(1), 33. Retrieved from https://doi.org/10.1007/s42979-022-01446-w doi: 10.1007/s42979-022-01446-w

[31] Sindhu, I., Daudpota, S. M., Badar, K., Bakhtyar, M., Baber, J., & Nurunnabi, M. (2019). Aspect-based opinion mining on student's feedback for faculty teaching performance evaluation. *IEEE Access*, *7*, 108729-108741.

[32] Syam Mohan E, R. S. (2021). Survey on aspect based sentiment analysis using machine learning techniques. *European Journal of Molecular amp; Clinical Medicine*, *7*(10), 1664-1684. Retrieved from https://ejmcm.com/article_6773.html

[33] WANJAWA, B., & MUCHEMI, L. (2020). Question answering using automatically generated semantic networks – the case of swahili questions. In *2020 ist-africa conference (ist-africa)* (p. 1-8).

[34] Xue, W., & Li, T. (2018, July). Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 2514–2523). Melbourne, Australia: Association for Computational Linguistics. Retrieved from https://aclanthology.org/P18-1234 doi: 10.18653/v1/P18-1234

[35] Zainuddin, N., Selamat, A., & Ibrahim, R. (2018). Hybrid sentiment classification on twitter aspect-based sentiment analysis. *Applied Intelligence*, *48*, 1218-1232.

[36] Zhang, B., Xu, X., Li, X., Chen, X., Ye, Y., & Wang, Z. (2019, sep). Sentiment analysis through critic learning for optimizing convolutional neural networks with rules. *Neurocomput.*, *356*(C), 21–30. Retrieved from https://doi.org/10.1016/j.neucom.2019.04.038 doi: 10.1016/j.neucom.2019.04.038

[37] Zhang, S., Hu, Y., & Bian, G. (2017). Research on string similarity algorithm based on levenshtein distance. In *2017 ieee 2nd advanced information technology, electronic and automation control*

conference (iaeac) (p. 2247-2251). doi: 10.1109/IAEAC.2017.8054419

[38] Zhou, G., Liu, Y., Liu, F., Zeng, D., & Zhao, J. (2013, 01). Improving question retrieval in community question answering using world knowledge. *IJCAI International Joint Conference on Artificial Intelligence*, 2239-2245.