Forum Text Processing and Summarization

Yen-Wei Mak and Hui-Ngo Goh

[1]Faculty of Computing and Informatics, Multimedia University, 63000 Cyberjaya, Selangor, Malaysia

Abstract. Finding answers from forums has been one of the most time consuming and tedious tasks for many people. This is because the answers are scattered across the forums and it is very hard to find the relevant answers. This problem particularly shows its face when it comes to software engineering. About 50 percent of searches don't provide enough results, according to Microsoft CEO Satya Nadella. This is especially true in software engineering, where it can be very challenging to zero down on the exact correct solution that solves your problem in such a sophisticated domain. Automatic FAQ Generation, Question Generation, and Answer Retrieval techniques arose over time, however most studies concentrated on broad topics like banking, healthcare, and so on. Therefore, in this work, we attempt to build upon prior research and transfer tried-and-true methods from other fields into the software engineering sphere. This study uses a number of natural language processing (NLP) methods, such as n-grams, to present a well-considered framework for achieving Automatic FAQ Generations and Retrieval from a software engineering perspective.

Keywords: Natural Language Processing (NLP), Deep Learning (DL), Summarization, Forum Processing, Sentiment Analysis

Hypothesis

Literature Review

This section provides an overview of the relevant existing research pertaining to the present study. The reviewed literature primarily revolves around FAQ Processing and Natural Language Processing (NLP) techniques.

Existing work in the domain of FAQ processing has predominantly centered on non-engineering fields, particularly the medical domain. However, the focus of this research is primarily directed towards the software engineering field. By exploring the existing findings, this paper aims to transfer any applicable insights derived from previous studies. Consequently, extensive reading has been undertaken to ensure proficiency in handling technical languages and terminologies.

FAQ retrieval plays a pivotal role in the ranking of question-answer pairs (Gupta & Carvalho, 2019). It involves the process of retrieving the most relevant questions from a large collection based on a user's query. Attaining high performance in this task would address several issues discussed in the previous Chapter 1.1.

However, traditional methods for FAQ generation heavily rely on extensive manual classification and software engineering techniques (Gupta & Carvalho, 2019). Such approaches demand significant time and effort to be executed. This concern has been underscored by previous studies conducted by Raazaghi (2015), Hu, Yu, and Jiau (2010), Henß, Monperrus, and Mezini (2012), and Razzaghi, Minaee, and Ghorbani (2016). Moreover, the quality of manual classification and software engineering also impacts the performance of the framework.

To address the aforementioned issues, various attempts and research have been undertaken to enhance the existing methods. One promising approach involves automating the process of FAQ generation using Natural Language Processing (NLP) techniques. Gupta and Carvalho (2019) have made notable advancements in this area by leveraging deep learning methods, specifically by combining Deep Matching Networks (DMN) and Multihop Attention Networks for FAQ retrieval.

Deep Matching Network (DMN) is a deep learning model that utilizes two matrix inputs to generate matching scores. These matrices are constructed by computing the dot product of word embeddings from both the questions and answers. The DMN has shown effectiveness in capturing semantic relationships between questions and answers (Gupta & Carvalho, 2019).

On the other hand, Multihop Attention Networks have demonstrated their efficacy in reasoning tasks such as question answering, which aligns with the focus of this paper (Gupta & Carvalho, 2019). This network incorporates multiple "hops" of attention to gather information from the input and make predictions. It involves an encoding step to encode the input and a decoder network that iteratively attends to different parts of the input, ultimately generating an output.

Raazaghi (2015) and Jijkoun and de Rijke (2005) proposed an approach that encompasses the entire architecture for achieving Auto-Faq-Gen. This architecture includes components such as web scraping, question construction, a ranking algorithm, and question generation. For a visual representation, please refer to Figure 2.1.

Open source forums serve as valuable repositories of information for users seeking answers to their questions. However, as time passes, these forums often become overwhelmed with a significant number of posts, making it challenging for users to navigate and find relevant information. Consequently, the implementation of a ranking mechanism within these forums can greatly enhance the user experience.

Hu et al. (2010) and Makino, Noro, and Iwakura (2016) have made notable strides in the area of selecting, weighing, clustering, and ranking contextual keywords. These advancements aim to achieve questions abstraction, thereby facilitating the process of locating pertinent questions and their corresponding answers within open source forums. The authors subsequently proposed a solution in the form of semi-automatic FAQ generation, which allows for improved organization and retrieval of information.

Neural Networks are a collection of algorithms inspired by the structure and functioning of the human brain. These algorithms enable the identification of patterns and possess the capability of independent learning. Through the evaluation of unstructured data, neural networks can adapt and adjust their weights to achieve optimal outcomes.

Over the past decade, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have emerged as prominent approaches for text processing tasks. In the study conducted by Duan, Tang, Chen, and Zhou (2017), CNN was proposed as a retrieval-based method for generating questions based on a given passage, while RNN was proposed as a generation-based approach. Both neural network models successfully demonstrated the ability to predict question patterns, select relevant question topics, rank questions, and ultimately generate questions. The proposed model outperformed traditional methods such as Mean Average Precision (MAP) or Mean Reciprocal Rank (MRR).

Another challenge arises when the questions posed by users cannot easily be classified to align with the existing questions in the FAQ database. This is often due to differences in form or context between the user's questions and the ones already present in the database (Makino et al., 2016). For instance, the user's question may be in a different language or may be expressed in a different format.

In addition to contextual variations, grammatical errors and misspellings pose further obstacles when developing automated question answering systems, as emphasized by Kothari, Negi, Faruquie, Chakaravarthy, and Subramaniam (2009). These linguistic challenges necessitate robust techniques to handle diverse language usage and to accurately interpret and respond to questions.

Context matching plays a crucial role in ranking the answers to user queries. However, the traditional approach of scoring similarity based on Levenshtein distance, as highlighted by S. Zhang, Hu, and Bian (2017), has limitations in effectively ranking answers. This is because Levenshtein distance fails to capture the semantic meaning of words, which is essential for accurate ranking.

Another concept worth considering is the Bag-of-Words (BOW) model, as discussed by Zhou, Liu, Liu, Zeng, and Zhao (2013). BOW similarity matching algorithms can be applied to processed text, including steps such as stop word removal and stemming, to calculate the similarity between the query and the answer. However, similar to Levenshtein distance, BOW is incapable of capturing the semantic meaning of words. Consequently, it can lead to false conceptual similarity between the query and the answer.

To overcome these limitations, more advanced techniques that consider semantic meaning, such as semantic matching models or neural network-based approaches, have been proposed in recent research to improve the accuracy of answer ranking in the context of user queries.

Word knowledge or word embedding offers a potential solution to improve traditional similarity matching algorithms. Zhou et al. (2013) proposed a method that incorporates word knowledge to enhance similarity matching. Their model connects a knowledge base to individual words, constructing a knowledge table that encompasses raw words, hypernyms, synonyms, and associative concepts. This approach deviates from traditional similarity matching algorithms by considering the semantic meaning of words rather than just the raw words themselves.

In the domain of question retrieval, Othman, Faiz, and Smaïli (2019) have made progress by introducing a method called "WEKOS" (Word Embedding, Kmeans, and COSine-based method). WEKOS leverages word embeddings and the cosine similarity metric to retrieve relevant questions. Specifically, the word embeddings of a given question are weighted and averaged to create an overall representation. Continuous word representations are learned in advance using the continuous bag-of-words (CBOW) model. The cosine similarity is then employed to calculate the similarity between the average word vectors of the query and those of existing questions.

These advancements in utilizing word knowledge and word embedding techniques offer promising approaches to enhance similarity matching and question retrieval processes.

Deep Matching Network (DMN) is a deep learning model that generates matching scores by leveraging two matrix inputs. These matrices are constructed by taking the dot product of the word embeddings of each word in the question with every word in the answers. The DMN captures the relationship between the question and answers through the matching scores it generates.

In contrast, the Multi-hop Attention Network has demonstrated effectiveness in reasoning tasks, specifically question answering, as highlighted by Gupta and Carvalho (2019). This network employs multiple "hops" of attention to gather information from the given input and make predictions. The process begins with encoding the input, and then a decoder network iteratively attends to different parts of the input, enabling the generation of an output.

These deep learning models, including the Deep Matching Network and the Multi-hop Attention Network, offer valuable techniques for capturing relationships and performing reasoning tasks, such as question answering, within the context of our research.

In their work, Makino et al. (2016) propose a method that utilizes a document classifier to predict whether a query corresponds to a frequently asked question (FAQ). The document classifier employs binary classifiers to classify each FAQ with respect to the query. To capture the dependency relation between the query and the FAQ, the sentences are divided into unigrams and bigrams.

Automatically Generated Semantic Networks refer to networks that are created through machine learning algorithms using text or other data sources. These networks serve to represent the structured relationships and connections between various concepts, entities, or topics. They are employed in natural language processing (NLP) and information retrieval tasks, such as classification, summarization, and question answering. Essentially, they can be considered as knowledge graphs constructed automatically from textual data.

In traditional Question Answering (QA) systems, syntactic and semantic methods are commonly employed. However, these methods may not perform effectively in low resource languages like Swahili. In such cases, semantic networks can offer an alternative solution, particularly in languages like Swahili, where the language structure generally follows a subject-verb-object pattern (WANJAWA & MUCHEMI, 2020).

Topic Modeling

Topic Modeling is a valuable approach for identifying question and answer pairings within a text corpus. It is a statistical modeling technique that aims to uncover the latent "topics" present in a collection of documents. By applying topic modeling, hidden semantic structures can be revealed, providing insights into the underlying themes within the text.

One commonly used topic modeling method is Latent Dirichlet Allocation (LDA). LDA is a generative probabilistic model that assumes each document is a mixture of topics, and each topic is a mixture of words. By training the model on a corpus of documents, LDA can discover the topics present in the collection. These topics can then be utilized to assign labels to new, unseen documents, enabling the identification of relevant question and answer pairings (Henß et al., 2012).

Henß et al. (2012) propose an architecture that leverages topic modeling in the prediction of question and answer pairings. Their proposed pipeline involves extracting topics from the corpus, which are then used to label the questions and answers. This labeling process facilitates the extraction of answers that closely align with the topics identified by the model.

Stop Words

Stop words are commonly encountered in natural language data and are typically filtered out during or after text processing. However, the specific set of stop words used can vary across different natural language processing tools, and there is no universal list applicable to all applications. Technical languages have their own unique set of stop words, which differs from the general stop words list used in applications like the NLTK library (Gerlach et al., 2019) (Sarica & Luo, 2020).

To address the lack of specific stop words for software engineering texts, Gerlach et al. (2019) developed a stop words list tailored to this domain. The list was created using statistical identification techniques and evaluated by domain experts.

The detection of phrases can be achieved using the algorithm proposed by Mikolov et al. (2013). This algorithm identifies frequently co-occurring words, allowing the detection of meaningful phrases within the text.

Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a process used to determine the sentiment expressed in a piece of writing, classifying it as positive, negative, or neutral. It is commonly employed to gain insights into people's attitudes and opinions about various topics. For example, sentiment analysis can be applied to assess public sentiment towards a new movie or to understand the overall sentiment towards a newly launched product.

While social media platforms like Twitter, Facebook, and Instagram are frequently used as sources for sentiment analysis, this technique can be applied to any text data. In the context of government entities, (Alqaryouti et al., 2019) utilize sentiment analysis to further investigate and comprehend the needs and preferences of customers.

Sentiment analysis can be performed at different levels of granularity, which refers to the level of detail at which sentiment is expressed. The three primary levels of granularity are sentence-level, document-level, and aspect-level. Considering the appropriate level of granularity is crucial as it impacts the specific type of sentiment analysis that can be conducted and the resulting insights obtained (Alqaryouti et al., 2019).

Abstractive Summarization

Abstractive summarization involves generating new sentences that capture the meaning of the original text. One common approach to abstractive summarization is using a sequence-to-sequence neural network, such as RNN or LSTM, which are well-suited for processing sequential data like text. The recurrent connections in these models enable them to maintain a hidden state that retains information from previous steps in the sequence, allowing them to capture contextual information as they read the input text. This enables the model to generate concise and semantically relevant summaries (Sharma & Sharma, 2022).

AMR (Abstract Meaning Representation) is an RNN-based method introduced by Banarescu et al. (2013). It utilizes a neural network model that produces a single graph representing time series information in the text to generate abstractive summaries.

Another approach is the ATSDL (Abstractive Text Summarization with Dual Learning) model, which combines the advantages of extractive and abstractive summarization. In this model, a sequence-to-sequence neural network is used to generate the summary, and then a sentence ranking model is employed to rank the sentences in the generated summary. The top-ranked sentences are selected to form the final summary (Sharma & Sharma, 2022).

To address the issue of unusual terms in abstractive models, the MOSP (Model for Open-domain Summarization with Phrase Representations) introduces a phrase collection model. This model generates and learns representations of phrases and their relationships within a document, thereby tackling the problem of handling unusual terms that abstractive models often encounter.

Methodology

The framework presented encompasses a series of operations designed to be applied to the data. Figure 3.1 visually represents the entirety of the framework, illustrating its components and their interactions.

The proposed pipeline incorporates all the essential components necessary to comprehensively address the required procedures for solving the problem at hand.

Assumptions

The titles of posts typically exhibit brevity and conciseness, serving as the initial focal point for individuals seeking solutions to their inquiries. Consequently, it is reasonable to presume that the title effectively encapsulates the essence of the corresponding post. In light of this, it is imperative to consider that all comments and answers should be contained within the body of the post itself, as it is the comprehensive representation of the information. When implementing a ranking system, it is essential to evaluate the post as a unified entity, taking into account both its content and associated responses.

1. Data Organization

In the process of data organization, the information will be categorized into distinct entities based on their types, namely posts, comments, answers, and answer comments. This categorization serves the purpose of enhancing data management and facilitating subsequent analyses. It is necessary to undertake this step due to the inherent lack of organization in the provided CSV dataset, which hinders efficient data handling. A visual representation depicting the comparison between the unorganized and organized data is presented in the figure xxx below.

2. Post Ranking - String Matching

Within the string matching framework, two widely used methods are FuzzyWuzzy and spaCy. FuzzyWuzzy is a battle-tested, state-of-the-art model for string matching, known for its remarkable performance in approximate and partial string matching tasks. It leverages sophisticated techniques such as Levenshtein distance calculations.

FuzzyWuzzy employs two methods: the partial ratio and token sort ratio. The partial ratio method measures the similarity between two strings by evaluating the ratio of the longest contiguous matching substrings. It effectively handles cases where matching substrings, rather than individual characters, play a crucial role. On the other hand, the token sort ratio method focuses on sorting the tokens within each string alphabetically and computing the similarity ratio based on the sorted token lists. This approach proves advantageous when comparing strings with varying word order, as it captures similarities that may be obscured by different word arrangements.

It is important to note that while both FuzzyWuzzy methods demonstrate robustness in various scenarios, they do not explicitly consider the contextual meaning of the strings being compared. To further enhance the string similarity analysis and capture the semantic relationships between strings, this study proposes the integration of RoBERTa-based entailment. RoBERTa is a powerful pretrained model that has been fine-tuned on the MNLI (Multi-Genre Natural Language Inference) dataset.

Entailment, in the context of string matching, refers to the relationship between two texts where one text logically follows from or can be inferred from the other. By utilizing RoBERTa, we can leverage its fine-tuned knowledge to determine the likelihood of one string entailing or implying the other. This approach allows us to capture not only surface-level similarities but also the underlying meaning and context of the strings being compared.

In practical terms, we utilize the "roberta-large-mnli" pretrained model, which has been specifically trained to perform natural language inference tasks. Given two input texts, we encode them using the model's tokenizer. Then, we pass the encoded texts through the RoBERTa model to obtain the logits, representing the probabilities of different entailment labels. By applying a softmax function to these logits, we obtain the probability distribution over the entailment labels. Finally, we extract the entailment probability for the "ENTAILMENT" label, which indicates the likelihood of the two texts being entailed.

By incorporating RoBERTa-based entailment into the string matching process, we enrich the analysis by considering not only surface-level similarities but also the semantic relationships and logical implications between strings. This approach enhances the accuracy and precision of the matching process, as it captures a deeper understanding of the strings' contextual meaning.

3. Preprocessing - Finding tech related stop words

The subsequent step entails identifying technology-related stop words to facilitate the removal of these words from the data. Stop words are commonly occurring words that do not hold significant importance for the analysis and are often used frequently in the text.

Traditionally, the process of finding stop words has been a laborious task, involving the utilization of manually curated stop word lists (Gerlach et al., 2019). These lists can be sourced from various origins, such as domain-specific stop word lists or language-specific stop word lists. However, in the case of our study, which focuses on the software engineering domain, conventional stop word lists are inadequate as they lack specificity to this particular domain. Consequently, employing such general stop word lists could introduce noise into the dataset.

To address this challenge, (Sarica & Luo, 2020) have made significant contributions by curating a stop word list tailored specifically to the software engineering domain. Their approach involved thorough analysis of data extracted from patent documents, which predominantly describe domains related to software engineering. Notably, they employed a range of techniques, as depicted in Figure 3.9, including preprocessing methods, a ranking framework based on term statistics, and an evaluation conducted by domain experts on a term-by-term basis. The meticulously curated stop word list developed by (Sarica & Luo, 2020) is utilized in this study, ensuring its relevance and suitability for our specific research objectives.

It is important to acknowledge that alternative methods, such as employing word clouds to identify the most frequent words in the data and subsequently removing them, exist for deriving stop word lists. However, such approaches may lack precision as they do not consider the contextual nuances of the data or the significance of individual words within the dataset. Therefore, the comprehensive approach proposed by (Sarica & Luo, 2020) emerges as a more effective solution, incorporating domain-specific considerations and expert evaluation to curate the appropriate stop word list for the software engineering domain.

4. Preprocessing - Remove Special Characters

Following the identification of special characters present in the dataset scraped from StackOverflow, the subsequent step involves their removal. The dataset contains a variety of special characters, and it is crucial to address this issue as these characters can lead to undesired complications during analysis. Table 3.1 provides an overview of the special characters identified within the dataset.

Removing special characters is necessary to ensure data cleanliness and facilitate subsequent processing and analysis tasks. By eliminating these characters, the dataset becomes more standardized and amenable to further analysis. The removal of special characters is typically achieved through text preprocessing techniques, such as pattern matching and substitution, that target the specific characters identified in Table 3.1. This process effectively cleanses the data by eliminating the unwanted special characters, thereby enhancing the quality and reliability of subsequent analyses.

5. Preprocessing - Extract URL from text

In order to leverage the valuable information embedded in URLs and understand the context of the data, it is crucial to incorporate them into the dataset. URLs can provide additional insights and references related to the data being analyzed. Since the data is scraped and contains HTML tags, extracting the URLs can be easily achieved using the BeautifulSoup package.

By employing the capabilities of the BeautifulSoup package, the URLs present within the data can be extracted effectively. This allows for the isolation of the URLs from the rest of the text, enabling their separate storage in a dedicated column within the dataset. This organization facilitates easy access to the URLs for future reference and analysis.

By storing the URLs in a separate column, the dataset maintains its structural integrity and allows for the establishment of connections between the data and the associated URLs. This integration of URLs provides researchers with the ability to explore additional information and enrich the understanding of the underlying context within the dataset.

6. Preprocessing - Removing code blocks

The framework recognizes the significance of removing code blocks from the data. While code blocks may not be a major concern in typical natural language processing tasks, they are prevalent in the context of Stack Overflow, which serves as a platform for developers to seek assistance and share their programming knowledge. Given the specific focus on Stack Overflow data, the removal of code blocks becomes a crucial step in the preprocessing process to minimize noise and enhance the quality of the dataset.

Code blocks within the data are typically enclosed within triple quotes ("```" or """""). This distinctive pattern simplifies the identification of code blocks within the pipeline, making it straightforward to recognize and subsequently eliminate them from the dataset. By removing code blocks, the framework aims to refine the dataset and ensure that the subsequent analyses and modeling efforts are not influenced by the presence of code snippets.

7. Preprocessing - Remove HTML Tags

The presence of HTML tags in the data is a significant concern that requires careful attention and removal during the preprocessing phase. The scraping process involved in collecting the data may inadvertently result in the inclusion of HTML tags within the textual content. By leveraging the distinctive pattern of HTML tags enclosed within angle brackets ("<" and ">"), the framework can readily identify and remove these tags. This step is essential to eliminate potential interference, improve data integrity, and enhance readability for subsequent text processing and analysis tasks.

8. Preprocessing - Lemmatization/Stemming

In the field of Natural Language Processing, text normalization techniques such as stemming and lemmatization are employed to prepare sentences, words, and documents for analysis. These techniques aim to reduce words to their root or base form. For example, the terms "kick" and "kicked" both stem from the verb "to kick," and it is desirable for a natural language processing application to recognize this relationship.

Stemming and lemmatization can be implemented using popular Python libraries like NLTK. However, the choice between stemming and lemmatization is a subject of debate. Stemming is a simple heuristic approach that truncates word endings to achieve the desired goal in most cases, often involving the removal of derivational affixes. On the other hand, lemmatization is a more sophisticated technique that considers morphological analysis and utilizes a lexicon to identify the base or dictionary form of a word, known as the lemma. It focuses on removing only inflectional endings.

9. Sentence Scoring

To improve the effectiveness of information retrieval systems, we propose a fuzzy time weightage scoring mechanism that incorporates fuzzy matching, time relevance, and vote count. The goal is to rank titles based on their relevance to a given query while considering both textual similarity and temporal proximity.

The mechanism begins by utilizing the fuzzy wuzzy method, a popular string matching technique, to calculate a similarity score between the query and each title. This score captures the textual similarity, accounting for variations and discrepancies between the query and the title text.

Formula for Fuzzy Matching Score:

$$\text{Similarity Score} = \text{fuzzy\_match}(\text{query}, \text{title})$$

In addition to the fuzzy matching score, the mechanism incorporates the time relevance of each title. The created date of each title is compared to the current time, and a time difference score is calculated. This score represents the temporal proximity of the title to the present moment. The closer the created date is to the current time, the higher the time score assigned to the title.

Formula for Time Score:

$$\text{Time Score} = 100 - ((\text{current\_time} - \text{created\_date}) / \text{max\_time\_diff}) * 100$$

Furthermore, the mechanism considers the vote count of each title as an indicator of its popularity or relevance. The vote count is transformed into a vote count score, taking into account the minimum and maximum vote counts in the dataset. The score is calculated as a percentage of the vote count's position within the vote count range, ensuring that higher vote counts receive higher scores.

Formula for Vote Count Score:

$$\text{Vote Count Score} = 100 - ((\text{vote\_count} - \text{min\_vote\_count}) / (\text{max\_vote\_count} - \text{min\_vote\_count})) * 100$$

To achieve a balanced ranking, weightages are assigned to each score component. In our approach, the fuzzy matching score carries a weightage of 80%, reflecting its primary importance in capturing textual similarity. The time score and vote count score contribute with weightages of 10% each, acknowledging their relevance but to a lesser degree compared to textual similarity.

Formula to Calculate Weighted Score:

$$\text{Weighted Score} = (\text{fuzzy\_weightage} * \text{Similarity Score}) + (\text{time\_weightage} * \text{Time Score}) + (\text{vote\_weightage} * \text{Vote Count Score})$$

The final weighted score for each title is obtained by combining the fuzzy matching score, time score, and vote count score according to their respective weightages. The mechanism sorts the titles based on the weighted scores in descending order, ensuring that titles with higher overall scores are ranked higher in the retrieval results.

Experimental evaluation of the fuzzy time weightage scoring mechanism demonstrates its effectiveness in improving the retrieval accuracy. By considering both textual similarity and temporal proximity, as well as incorporating the popularity aspect through the vote count, the mechanism provides more relevant and timely results to users.

In conclusion, the fuzzy time weighted scoring mechanism offers a comprehensive approach to information retrieval, incorporating fuzzy matching, time relevance, and vote count. By striking a balance between textual similarity, recency, and popularity, the mechanism enhances the effectiveness of retrieval systems and provides users with more accurate and timely results.

10. Sentiment analysis

Sentiment analysis, also known as opinion mining, is a crucial process in determining the sentiment or attitude expressed in a piece of writing, whether it is positive, negative, or neutral. In the context of our framework, sentiment analysis plays a pivotal role in enhancing the ranking of posts.

Upon scoring the posts using the aforementioned sentence scoring method and assessing their relevance, it becomes apparent that the scores alone may not offer sufficient information to judge the usefulness of a post. To overcome this limitation, sentiment analysis is introduced as an additional factor in the ranking process.

By conducting sentiment analysis, the assumption is made that posts with a positive sentiment are more likely to be useful compared to those with a negative sentiment. This assumption allows for the incorporation of sentiment analysis as an additional layer of filtering within the ranking aspect of the framework.

Initially, the Twitter RoBERTa base sentiment analysis model was employed, given its popularity with over 2 million applications this month. This model is based on RoBERTa, a widely-utilized transformer-based model, and has been trained on an extensive dataset encompassing approximately 124 million tweets from January 2018 to December 2021. However, subsequent research revealed a superior alternative.

In the latest study, the Senti4SD model was utilized, surpassing the performance of the Twitter RoBERTa base sentiment analysis model significantly. Senti4SD is an emotion polarity classifier specifically designed for sentiment analysis in developers' communication channels. Training and evaluation of this model were conducted using a gold standard dataset comprising over 4,000 posts extracted from Stack Overflow. It is a component of the Collab Emotion Mining Toolkit (EMTk), catering to sentiment analysis requirements in software development contexts.

The Senti4SD model demonstrates remarkable performance by accurately predicting sentiment and providing probability scores for three sentiment classes: positive, negative, and neutral. Leveraging the capabilities of this model allows for the determination of sentiment expressed in a given text, with corresponding probability scores assigned to each sentiment class. Training on a domain-specific dataset from Stack Overflow enhances its effectiveness in capturing sentiments prevalent in developers' communication channels.

The incorporation of the Senti4SD model into the framework elevates the accuracy and reliability of the sentiment analysis process. By considering the sentiment expressed in each post alongside the sentence scores, a comprehensive understanding of the posts' usefulness and relevance is obtained. This refined approach enables a more precise ranking of the posts within the framework, facilitating improved decision-making based on sentiment analysis.

11. Summarization

As the framework reaches the final stage of the pipeline, it aims to enhance the user experience by providing a summary of the top 5 ranked posts. Summarization is a process that involves condensing a text document using software to create a concise summary that captures the key points of the original document. The goal of summarization is to reduce the length of the text while preserving the most important information.

In our framework, summarization plays a critical role as it enables users to comprehend the posts more efficiently, quickly, and effortlessly. By generating summaries of the top 5 sentiment-ranked posts, we aim to facilitate a better understanding of the content and enable users to grasp the essential information more easily.

The chosen model for summarization is the widely used BART (Bidirectional and Autoregressive Transformer) large CNN model, which has been utilized more than 1 million times this month. BART is a transformer-based encoder-decoder model that combines bidirectional encoding (similar to BERT) and autoregressive decoding (similar to GPT). It was pretrained on English language data and fine-tuned on the CNN Daily Mail dataset.

To enhance the performance of the BART model for our specific use case, we conducted fine-tuning using the SOSUM dataset. This dataset consists of extractive summaries from 2,278 Stack Overflow (SO) posts related to 506 of the most popular SO questions. The purpose of this fine-tuning was to tailor the BART model to better understand and generate summaries specifically for Stack Overflow posts. In the final section of this paper, we will present and discuss the results of the fine-tuned BART model and compare its performance to the base model. This analysis will demonstrate how the fine-tuned model outperforms the original model, highlighting its effectiveness in generating more accurate and informative summaries.

For each of the top 5 sentiment-ranked posts, summarization will be applied. This includes summarizing the main post, post comments, answers, and answer comments. The sentences are sorted based on time to maintain chronological order, allowing BART to capture any time-related information and the context of ongoing conversations. This includes recognizing when users mention each other in the comments section.

Results and Discussion

In this section We conduct both quantitative experiments and user studies to answer the following four research questions:

1. How effective is our question matching, sentiment analysis and summarization model in software engineering domains?
2. Is the fine-tuned summarization model better than the base BART model?
3. How do real programmers perceive the summaries given by our models?

A. Datasets descriptions

It's important to note that all of the dataset picked are somewhat closely related to software engineering domains or at least have the same nature of technicality online forums. The dataset used to evaluate the **question matching methods** are the quora questions pair dataset, the dataset provided pairs of questions contain two questions with the same meaning. The ground truth is the set of labels that have been supplied by human experts. To evaluate the **Sentiment Analysis model**, the dataset chosen were …, the data are directly scraped from stackoverflow, thus making perfect sense to our research work. Lastly, the **summarization evaluation** datasets are from SOSUM, this dataset consists of extractive summaries from 2,278 Stack Overflow (SO) posts related to 506 of the most popular SO questions.

B. Evaluation Methods

In this section, the methods to evaluate the separate components are detailed and discussed. While the results of the individual components are noted in section c - results. It is also important to note that this section is directly related to the research questions above.

1. Question matching models

   4 different methods are tested, fuzzy wuzzy partial ratio, fuzzy wuzzy token sort, spacy similarity and the roberta-large-mnli model. All 4 methods are evaluated using the quora questions pair dataset, confusion matrices, F1, recall and precision scores are noted down to perform further analysis.

2. Sentiment analysis models

The roberta model and senti4sd model are both tested with 2 sets of datasets. Similarly, confusion matrices, F1, recall and precision scores are detailed below.

3. Summarization models

As part of our research work, the paper fine-tuned a base BART model with the SOSUM datasets, to further improve its accuracy. Both of these models are evaluated with the SOSUM model. It is important to note that the datasets used to fine tune the BART model are splitted to training datasets and evaluations datasets whereby the training data is fed into the model while fine tuning and the evaluation datasets are treated as unseen data.

4. Verdict by real programmers

Apart from model evaluations, the paper focuses on combining all models together and ultimately proposing a framework to help developers filter, analyze, and summarize online forums. It goes without saying that a human evaluation is a must to test the entire framework. The research team behind this paper, generated 10 pairs of question and answers, this involves generating 10 queries and feeding to the model to infer the best stackoverflow posts. The experiment is set up whereby the original post and the summarized content of the post is both presented to the testers. 10 undergraduate students are then picked from our institute to perform human evaluations. The testers will then note down if the ranked posts helped them to find answers easier and if the summarized output is beneficial to them in terms of understanding the answers.

C. Results

In this section, we detailed down the results of 4 of the evaluation methods we've proposed in the previous section.

1. Question Matching Models

	Precision	Recall	F1
Fuzzy wuzzy partial ratio	30.03%	52.61%	38.23%
Fuzzy wuzzy token sort	29.10%	56.12%	38.33%
Spacy Similarity	96.35%	40.24%	56.77%
Roberta Large MNLI Entailment	76.70%	66.63%	71.31%

2. Sentiment Analysis Models

	Precision	Recall	F1

Roberta			
Senti 4SD	60%	100%	74.90%

3. Summarization Models

   There's 3 evaluation methods in ROUGE that are used in our evaluation process. Rouge-1 evaluates individual words, Rouge-2 assesses word pairs, and Rouge-L considers overall structure and content overlap. Using a combination of metrics provides a comprehensive evaluation. The suitability of each metric varies based on specific evaluation needs. Rouge-1 is useful for keyword accuracy, Rouge-2 focuses on cohesion and fluency, and Rouge-L allows flexibility in word order and sentence structure. By considering multiple metrics, a broader range of summary qualities can be assessed, offering a more holistic view of system performance. Ultimately, the choice should align with the goals and requirements of the summarization task.

   Rouge-1

	Precision	Recall	F1
Base BART	22.40%	83.24%	34.43%
Fine-tuned BART	46.01%	75.36%	53.47%

   Rouge-2

	Precision	Recall	F1
Base BART	14.70	70.95%	23.69%
Fine-tuned BART	40.32%	69.85%	47.12%

   Rouge-L

	Precision	Recall	F1
Base BART	21.88%	81.41%	33.64%
Fine-tuned BART	45.72%	74.96%	53.16%

4. Verdict by real programmers

Question	Not relevant	Relevant	Preferred Ranking	Preferred Sumarization
1				

2				
3				
4				
5				
6				
7				
8				
9				
10				